Arthur-Merlin Streaming Complexity

Tom Gur * Ran Raz *

February 5, 2013

Abstract

We study the power of Arthur-Merlin probabilistic proof systems in the data stream model. We show a canonical $\mathcal{AM}$ streaming algorithm for a wide class of data stream problems. The algorithm offers a tradeoff between the length of the proof and the space complexity that is needed to verify it.

As an application, we give an $\mathcal{AM}$ streaming algorithm for the *Distinct Elements* problem. Given a data stream of length m over alphabet of size n, the algorithm uses $\tilde{O}(s)$ space and a proof of size $\tilde{O}(w)$, for every s, w such that $s \cdot w \geq n$ (where $\tilde{O}$ hides a $\text{polylog}(m, n)$ factor). We also prove a lower bound, showing that every $\mathcal{MA}$ streaming algorithm for the *Distinct Elements* problem that uses s bits of space and a proof of size w, satisfies $s \cdot w = \Omega(n)$.

As a part of the proof of the lower bound for the *Distinct Elements* problem, we show a new lower bound of $\Omega\left(\sqrt{n}\right)$ on the $\mathcal{MA}$ communication complexity of the *Gap Hamming Distance* problem, and prove its tightness.

Keywords: Probabilistic Proof Systems, Data Streams, Communication Complexity.

1 Introduction

The data stream computational model is an abstraction commonly used for algorithms that process network traffic using sublinear space [AMS96, IW03, CCM09]. In the settings of this model, we have an algorithm that gets a sequence of elements (typically, each element is an integer) as input. This sequence of elements is called a *data stream* and is usually denoted by $\sigma = (a_1, \ldots, a_m)$; where a_1 is the first element, a_2 is the second element, and so forth. The algorithm receives its input (a data stream) element-by-element. After it sees each a_i, it no longer has an access to elements with index that is smaller than i. The algorithm is required to compute a function of the data stream, using as little space as possible.

*Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: {tom.gur, ran.raz}@weizmann.ac.il. Research supported by an Israel Science Foundation grant and by the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation.

Among the most fundamental problems in the data stream model is the problem of *Distinct Elements*, i.e., the problem of computing the number of distinct elements in a given data stream. The problem has been studied extensively in the last two decades (see, for example, [AMS96, IW03, KNW10]). Its significance stems both from the vast variety of applications that it spans (covering IP routing, database operations and text compression, cf. [Mut05, AMS96, GKG05]), and due to the theoretical insight that it gives on the nature of computation in the data stream model.

Alon at el. [AMS96] have shown a lower bound of $\Omega(n)$ (where n is the size of the alphabet from which the elements are taken) on the streaming complexity of the computation of the *exact* number of distinct elements in a sufficiently long data stream (i.e., where the length of the data stream is at least proportional to n). The goal of reducing the space complexity of the *Distinct Elements* problem has led to a long line of research of approximation algorithms for the problem, starting with the seminal paper [FM83] by Flajolet and Martin. Recently, Kane at el. [KNW10] gave the first optimal approximation algorithm for estimating the number of distinct elements in a data stream; for a data stream with alphabet of size n, given $\epsilon > 0$ their algorithm computes a $(1 \pm \epsilon)$ multiplicative approximation using $O(\epsilon^{-2} + \log n)$ bits of space, with 2/3 success probability.

A natural approach for reducing the space complexity of streaming algorithms, *without* settling on an approximation, is by considering a probabilistic proof system. Chakrabarti at el. [CCM09] have shown *data stream with annotations* algorithms for several data stream problems, using a probabilistic proof system that is very similar to $\mathcal{MA}$. This line of work continued in [CMT10], wherein a probabilistic proof system was used in order to reduce the streaming complexity of numerous graph problems. In a subsequent work [CMT11], Chakrabarti at el. provided a practical instantiation of one of the most efficient general-purpose construction of an interactive proof for arbitrary computations, due to Goldwasser at el. [GKR08].

In this work, we study the power of Arthur-Merlin probabilistic proof systems in the data stream model. We show a canonical $\mathcal{AM}$ streaming algorithm for a wide class of data stream problems. The algorithm offers a tradeoff between the length of the proof and the space complexity that is needed to verify it. We show that the problem of *Distinct Elements* falls within the class of problems that our canonical algorithm can handle. Thus, we give an $\mathcal{AM}$ streaming algorithm for the *Distinct Elements* problem. Given a data stream of length m over alphabet of size n, the algorithm uses $\tilde{O}(s)$ space and a proof of size $\tilde{O}(w)$, for every s, w such that $s \cdot w \geq n$ (where $\tilde{O}$ hides a polylog(m, n) factor).

In addition, we give a lower bound on the $\mathcal{MA}$ streaming complexity of the *Distinct Elements* problem. Our lower bound for *Distinct Elements* relies on a new lower bound that we prove on the $\mathcal{MA}$ communication complexity of the *Gap Hamming Distance* problem.

1.1 Arthur-Merlin Probabilistic Proof Systems

An $\mathcal{MA}$ (Merlin-Arthur) proof is a probabilistic extension of the notion of proof in complexity theory. Proofs of this type are commonly described as an interaction between two players, usually referred to as Merlin and Arthur. We think of Merlin as an omniscient prover, and

of Arthur as a computationally bounded verifier. Merlin is supposed to send Arthur a valid proof for the correctness of a certain statement. After seeing both the input and Merlin's proof, with high probability Arthur can verify a valid proof for a correct statement, and reject every possible alleged proof for a wrong statement.

Formally, the complexity class $\mathcal{MA}(T, W)$ is defined as follows:

Definition 1.1. *Let $\epsilon \geq 0$, and let $T, W : \mathbb{N} \to \mathbb{N}$ be monotone functions. A language L is in $\mathcal{MA}_\epsilon(T, W)$ if there exists a randomized algorithm V (the verifier) that receives an input x (denote its size by $|x|$) and a proof (sometimes called witness) w, such that,*

1. ***Completeness**: For every $x \in L$, there exists a string w of length at most $W(|x|)$ that satisfies*

$$\Pr[V(x, w) = 1] > 1 - \epsilon.$$

2. ***Soundness**: For every $x \notin L$, and for any string w of length at most $W(|x|)$,*

$$\Pr[V(x, w) = 1] < \epsilon.$$

3. *For every x, w the running time of V on (x, w) is at most $T(|x|)$.*

Under these notations, we refer to T as the time complexity *of the verifier. The function W is referred to as the* length *of the proof, and the sum $T + W$ is called the $\mathcal{MA}$ complexity of the algorithm.*

An $\mathcal{AM}$ proof is defined almost the same as an $\mathcal{MA}$ proof, except that in $\mathcal{AM}$ proof systems we assume that both the prover and the verifier have access to a common source of randomness (alternatively, $\mathcal{AM}$ proof systems can be described as $\mathcal{MA}$ proof systems that start with an extra round, wherein Arthur sends Merlin a random string).

The notion of $\mathcal{AM}$ and $\mathcal{MA}$ proof systems can be extended to many computational models. In this work we consider both the communication complexity analogue of $\mathcal{MA}$, wherein Alice and Bob receive a proof that they use in order to save communication, and the data stream analogues of $\mathcal{MA}$ and $\mathcal{AM}$, wherein the data stream algorithm receives a proof and uses it in order to reduce the required resources for solving a data stream problem.

Recently, probabilistic proof systems for streaming algorithms have been used to provide an abstraction of the notion of *delegation of computation* to a cloud (see [CMT10, CMT11, CKLR11]). In the context of cloud computing, a common scenario is one where a user receives or generates a massive amount of data, which he cannot afford to store locally. The user can stream the data he receives to the cloud, keeping only a short certificate of the data he streamed. Later, when the user wants to calculate a function of that data, the cloud can perform the calculations and send the result to the user. However, the user cannot automatically trust the cloud (as an error could occur during the computation, or the service provider might not be honest). Thus the user would like to use the short certificate that he saved in order to verify the answer that he gets from the cloud.

1.2 Communication Complexity and the *Gap Hamming Distance* Problem

Communication complexity is a central model in computational complexity. In its basic setup, we have two computationally unbounded players, Alice and Bob, holding (respectively) binary strings x, y of length n each. The players need to compute a function of both of the inputs, using the least amount of communication between them.

In this work we examine the well known communication complexity problem of *Gap Hamming Distance* (GHD), wherein each of the two parties gets an n bit binary string, and together the parties need to tell whether the Hamming distance of the strings is larger than $\frac{n}{2} + \sqrt{n}$ or smaller than $\frac{n}{2} - \sqrt{n}$ (assuming that one of the possibilities occurs). In [CR11] a tight linear lower bound was proven on the communication complexity of a randomized communication complexity protocol for GHD. Following [CR11], a couple of other proofs ([Vid11, She11]) were given for the aforementioned lower bound. Relying on [She11], in this work we give a tight lower bound of $\Omega(\sqrt{n})$ on the $\mathcal{MA}$ communication complexity of GHD.

1.3 Our Results

The main contributions in this work are:

1. A canonical $\mathcal{AM}$ streaming algorithm for a wide class of data stream problems, including the *Distinct Elements* problem.
2. A lower bound on the $\mathcal{MA}$ streaming complexity of the *Distinct Elements* problem.
3. A tight lower bound on the $\mathcal{MA}$ communication complexity of the *Gap Hamming Distance* problem.

In order to state the results precisely, we first introduce the following notations: given a data stream $\sigma = (a_1, \ldots, a_m)$ (over alphabet $[n]$), the *element indicator* $\chi_i : [n] \to \{0,1\}$ of the i'th element ($i \in [m]$) of the stream σ, is the function that indicates whether a given element is in position $i \in [m]$ of σ, i.e., $\chi_i(j) = 1$ if and only if $a_i = j$. Furthermore, let $\chi : [n] \to \{0,1\}^m$ be the *element indicator* of σ, defined by

$$\chi(j) = \big(\chi_1(j), \ldots, \chi_m(j)\big).$$

In addition, given $n \in \mathbb{N}$ we define a *clause* over n variables $x_1, \ldots, x_n$ as a function $C : \{0,1\}^n \to \{0,1\}$ of the form $(y_1 \vee y_2 \vee \ldots \vee y_n)$, where for every $i \in [n]$ the literal y_i is either a variable (x_j), a negation of a variable $(\neg x_j)$, or one of the constants $\{0,1\}$.

Equipped with the notations above, we formally state our results. Let $0 \leq \epsilon < 1/2$. Let $\mathcal{P}$ be a data stream problem such that for every $m, n \in \mathbb{N}$ there exists a set of $k = k(m,n)$ clauses $\{C_t\}_{t \in [k]}$ over m variables, and a function $\psi : \{0,1\}^k \to \mathbb{Z}$, such that for every data stream $\sigma = (a_1, \ldots, a_m)$ with alphabet $[n]$,

$$(1-\epsilon)\mathcal{P}(\sigma) \leq \sum_{j=1}^{n} \psi\big(C_1 \circ \chi(j), \ldots, C_k \circ \chi(j)\big) \leq (1+\epsilon)\mathcal{P}(\sigma).$$

Moreover, we assume that ψ and $\{C_t\}_{t\in[k]}$ are known to the verifier[1], and that there exists $B \leq \text{poly}(m,n)$ such that $\psi(x) < B$ for every $x \in \{0,1\}^k$. Given such $\mathcal{P}$, for every $0 < \delta \leq 1$ and every $s, w \in \mathbb{N}$ such that $s \cdot w \geq n$, we give an $\mathcal{AM}$ streaming algorithm, with error probability δ, for approximating $\mathcal{P}(\sigma)$ within a multiplicative factor of $1 \pm \epsilon$. The algorithm uses space $O\big(sk \cdot \text{polylog}(m,n,\delta^{-1})\big)$, a proof of size $W = O\big(wk \cdot \text{polylog}(m,n,\delta^{-1})\big)$, and randomness complexity $\text{polylog}(m,n,\delta^{-1})$.

We show that the aforementioned algorithm, when applied to the *Distinct Elements* problem with parameters s, w such that $s \cdot w \geq n$, yields an $\mathcal{AM}$ streaming algorithm for the problem. The algorithm computes, with probability at least $2/3$, the exact number of distinct elements in a data stream of length m over alphabet $[n]$, using space $\tilde{O}(s)$ and a proof of size $\tilde{O}(w)$ (where $\tilde{O}$ hides a $\text{polylog}(m,n)$ factor). For example, by fixing $w = n$, we get an $\mathcal{AM}$ streaming algorithm for the *Distinct Elements* problem that uses only polylogarithmic space.

We note that an interesting special case of the class of problems that our canonical $\mathcal{AM}$ streaming algorithm handles can also be stated in terms of Boolean circuits, instead of clauses. That is, given $0 \leq \epsilon < 1/2$ and a data stream problem $\mathcal{P}$ such that for every $m, n \in \mathbb{N}$ there exists an unbounded fan-in Boolean circuit $C : \{0,1\}^m \to \{0,1\}$ with $k = k(m,n)$ non-input gates, such that for every data stream $\sigma = (a_1, \ldots, a_m)$ with alphabet $[n]$,

$$(1-\epsilon)\mathcal{P}(\sigma) \leq \sum_{j=1}^{n} C\big(\chi_1(j), \ldots, \chi_m(j)\big) \leq (1+\epsilon)\mathcal{P}(\sigma).$$

Assuming that C is known to the verifier, we get an $\mathcal{AM}$ streaming algorithm for $\mathcal{P}$ with the same parameters as in the original formulation of the canonical $\mathcal{AM}$ algorithm above.

Our next result is a lower bound on the $\mathcal{MA}$ streaming complexity of the *Distinct Elements* problem. We show that every $\mathcal{MA}$ streaming algorithm that approximates, within a multiplicative factor of $1 \pm 1/\sqrt{n}$, the number of distinct elements in a data stream of length m over alphabet $[n]$, using s bits of space and a proof of size w, must satisfy $s \cdot w = \Omega(n)$.

Last, we show a tight (up to a logarithmic factor) lower bound on the $\mathcal{MA}$ communication complexity of the *Gap Hamming Distance* problem. For every $\mathcal{MA}$ communication complexity protocol for GHD that communicates t bits and uses a proof of size w, we have $t \cdot w = \Omega(n)$. We prove the tightness of the lower bound by giving, for every $t, w \in \mathbb{N}$ such that $t \cdot w \geq n$, an $\mathcal{MA}$ communication complexity protocol for GHD, which communicates $O(t \log n)$ bits and uses a proof of size $O(w \log n)$.

1.4 Techniques

The main intuition behind our canonical $\mathcal{AM}$ streaming algorithm is based on the "algebrization" inspired communication complexity protocol of Aaronson and Wigderson [AW09]. However our proof is much more technically involved.

[1]For example, ψ and $\{C_t\}_{t\in[k]}$ can be $\text{polylog}(m,n)$-space uniform; that is, the *description* of ψ and $\{C_t\}_{t\in[k]}$ can be computed by a deterministic Turing machine that runs in $\text{polylog}(m,n)$ space.

In general, say we have a data stream problem $\mathcal{P}$ and two integers s, w such that $s \cdot w \geq n$. If there exists a low degree polynomial $g(x, y) : \mathbb{Z}^2 \to \mathbb{Z}$ (that depends on the input stream σ) and two domains $\mathcal{D}_w, \mathcal{D}_s \subseteq \mathbb{Z}$ of cardinality w, s (respectively) such that

$$\mathcal{P}(\sigma) = \sum_{x \in \mathcal{D}_w} \sum_{y \in \mathcal{D}_s} g(x, y),$$

then assuming we can efficiently evaluate g at a random point, by a straightforward adaptation of the [AW09] protocol to the settings of streaming algorithms, we obtain a simple $\mathcal{MA}$ streaming algorithm for $\mathcal{P}$.

However, in our case we can only express $\mathcal{P}(\sigma)$ as

$$\sum_{x \in \mathcal{D}_w} \sum_{y \in \mathcal{D}_s} \psi\big(C_1 \circ \tilde{\chi}(x, y), \ldots, C_k \circ \tilde{\chi}(x, y)\big),$$

where k is a natural number, $\{C_t\}_{t \in [k]}$ are clauses over m variables, $\psi : \{0,1\}^k \to \mathbb{Z}$ is a function over the hypercube, $\tilde{\chi} : \mathcal{D}_w \times \mathcal{D}_s \to \{0,1\}^m$ is the bivariate equivalent of the element indicator $\chi : [n] \to \{0,1\}^m$, and $\mathcal{D}_w, \mathcal{D}_s \subseteq \mathbb{Z}$ are domains of cardinality w, s (respectively).

The function $\psi\big(C_1 \circ \tilde{\chi}(x, y), \ldots, C_k \circ \tilde{\chi}(x, y)\big)$ is not a low degree polynomial. We would have liked to overcome this difficulty by using the approximation method of [Raz87, Smo87]. The latter allows us to have a low degree approximation of the clauses $\{C_t\}_{t \in [k]}$, such that with high probability (over the construction of the approximation polynomials) we can replace the clauses with low degree polynomials, without changing the output. The aforementioned randomized procedure comes at a cost of turning the $\mathcal{MA}$ streaming algorithm to an $\mathcal{AM}$ streaming algorithm.

Yet, the above does not sufficiently reduces the degree of $\psi\big(C_1 \circ \tilde{\chi}(x, y), \ldots, C_k \circ \tilde{\chi}(x, y)\big)$. This is due to the fact that the method of [Raz87, Smo87] results with approximation polynomials over a finite field of cardinality that is larger than $\mathcal{P}(\sigma)$. The degree of the approximation polynomials is close to the cardinality of the finite field, which in our case can be a large number ($\mathrm{poly}(m, n)$).

Instead we aim to apply the method of [Raz87, Smo87] to approximate

$$\{\mathcal{P}(\sigma) \pmod q\}_{q \in Q}$$

for a set Q of $\mathrm{polylog}(m, n)$ primes, each of size at most $\mathrm{polylog}(m, n)$. This way, each approximation polynomial that we get is over a finite field of cardinality $\mathrm{polylog}(m, n)$, and of sufficiently low degree. Then, we use the *Chinese Remainder Theorem* to extract the value of $\mathcal{P}(\sigma)$ from $\{\mathcal{P}(\sigma) \pmod q\}_{q \in Q}$.

Nonetheless, this is still not enough, as for every $q \in Q$ we want the answer to be the summation of the polynomial approximation of $\psi\big(C_1 \circ \tilde{\chi}(x, y), \ldots, C_k \circ \tilde{\chi}(x, y)\big) \pmod q$ over some domain $\mathcal{D}_w \times \mathcal{D}_s \subseteq \mathbb{Z}^2$ (where $|\mathcal{D}_w| = w$ and $|\mathcal{D}_s| = s$). Since the cardinality of the field $\mathbb{F}_q$ is typically smaller than w and s, we use an extension (of sufficient cardinality) of the field $\mathbb{F}_q$.

At each step of the construction, we make sure that we perserve both the restrictions that are imposed by the data stream model, and the conditions that are needed to ensure an efficient verification of the proof.

The idea behind our $\mathcal{AM}$ streaming algorithm for *Distinct Elements* is simply noting that we can indicate whether an element j appears in the data stream, by the disjunction of the element indicators of $j \in [n]$ in all of the positions of the stream (i.e., $\chi_1(j), \ldots, \chi_m(j)$). Then we can represent the number of distinct elements as a sum of disjunctions, and use the canonical $\mathcal{AM}$ streaming algorithm in order to solve the *Distinct Elements* problem.

As for the lower bound on the $\mathcal{MA}$ streaming complexity of the *Distinct Elements* problem, we start by establishing a lower bound on the $\mathcal{MA}$ communication complexity of the *Gap Hamming Distance* problem (GHD). A key element in the proof of the latter is based on Sherstov's recent result [She11] on the *Gap Orthogonality* problem (ORT) and its relation to GHD. Sherstov observed that the problem of *Gap Orthogonality* readily reduces to *Gap Hamming Distance* problem. Although at first glance it seems that the transition to ORT is of little substance, it turns out that Yao's corruption bound [Yao83] suits it perfectly. In fact, the corruption property for ORT is equivalent to the anti-concentration property of orthogonal vectors in the Boolean cube. Using this observation, we prove a lower bound on the $\mathcal{MA}$ communication complexity of ORT (following the method of [RS04]), which in turn, by the reduction from ORT to GHD, implies a lower bound on the $\mathcal{MA}$ communication complexity of GHD. Next we adapt the reduction that was implicitly stated in [IW03], and reduce the $\mathcal{MA}$ communication complexity problem of GHD to the $\mathcal{MA}$ problem of calculating the exact number of *Distinct Elements*.

1.5 Related Work

The data stream model has gained a great deal of attention after the publication of the seminal paper by Alon, Matias and Szegedy [AMS96]. In the scope of that work, the authors have shown a lower bound of $\Omega(n)$ (where n is the size of the alphabet) on the streaming complexity of *Distinct Elements* (i.e., the computation of the *exact* number of distinct elements in a data stream) where the length of the input is at least proportional to n.

Following [AMS96] there was a long line of theoretical research on the approximation of the *Distinct Element* problem ([BYJK$^+$02, IW03, BHR$^+$07, BC09, KNW10], see [Mut05] for a survey of earlier results). Finally, Kane at el. [KNW10] gave the first optimal approximation algorithm for estimating the number of distinct elements in a data stream; for a data stream with alphabet of size n, given $\epsilon > 0$ their algorithm computes a $(1 \pm \epsilon)$ multiplicative approximation using $O(\epsilon^{-2} + \log n)$ bits of space, with 2/3 success probability. This result matches the tight lower bound of Indyk and Woodruff [IW03].

In a recent sequence of works, the data stream model was extended to support several interactive and non-interactive proof systems [CCM09, CMT10, CKLR11]. The model of streaming algorithms with non-interactive proofs was first introduced in [CCM09] and extended in [CMT10, CMT11]. In [CCM09] the authors gave an optimal (up to polylogarithmic factors) *data stream with annotations* algorithm for computing the k'th frequency moment exactly, for every integer $k \geq 1$.

2 Preliminaries

2.1 Communication Complexity

Let X, Y, Z be finite sets, and let $f : X \times Y \to Z$ be a (possibly partial) function. In the *two-party probabilistic communication complexity model* we have two computationally unbounded players, traditionally referred to as Alice and Bob. Both players share a random string. Alice gets as an input $x \in X$. Bob gets as an input $y \in Y$. At the beginning, none of the players has any information regarding the input of the other player. Their common goal is to compute the value of $f(x, y)$, using a protocol that communicates as small number of bits as possible. In each step of the protocol, one of the players sends one bit to the other player. This bit may depend on the player's input, the common random string, as well as on all previous bits communicated between the two players. At the end of the protocol, both players have to know the value of $f(x, y)$ with high probability.

2.1.1 MA Communication Complexity

In $\mathcal{MA}$ *communication complexity protocols*, we have a (possibly partial) function $f : X \times Y \to \{0, 1\}$ (for some finite sets X, Y), and three computationally unbounded parties: Merlin, Alice, and Bob. The function f is known to all parties. Alice gets as an input $x \in X$. Bob gets as an input $y \in Y$. Merlin sees both x and y. We think of Merlin as a *prover*, and think of Alice and Bob as *verifiers*. We assume that Alice and Bob share a private random string that Merlin cannot see.

At the beginning of an $\mathcal{MA}$ *communication complexity protocol*, Merlin sends a proof string w to both Alice and Bob, so both players have a free access to w. The players proceed as before. In each step of the protocol, one of the players sends one bit to the other player. At the end of the protocol, both players have to know an answer z. Hence, the answer depends on the input (x, y) as well as on the proof w. For a protocol P, denote by $P\big((x, y), w\big)$ the probabilistic answer z given by the protocol on input (x, y) and proof w.

An $\mathcal{MA}$ *communication complexity protocol* has three parameters: a limit on the probability of error of the protocol, denoted by ϵ; a limit on the number of bits of communication between Alice and Bob, denoted by T; and a limit on the length of Merlin's proof string, denoted by W.

With the above in mind, we can now define $\mathcal{MA}_\epsilon(T, W)$ communication complexity as follows:

Definition 2.1. *An $\mathcal{MA}_\epsilon(T, W)$-communication complexity protocol for f is a probabilistic communication complexity protocol P, as above (i.e., with an additional proof string w presented to the players). During the protocol, Alice and Bob communicate at most T bits. The protocol satisfies,*

1. ***Completeness**: for all $(x, y) \in f^{-1}(1)$, there exists a string w such that $|w| < W$,*

that satisfies

$$\Pr\left[P\big((x,y),w\big)=1\right]>1-\epsilon.$$

2. ***Soundness**: for all $(x,y)\in f^{-1}(0)$ and for any string w such that $|w|<W$, we have*

$$\Pr\left[P\big((x,y),w\big)=1\right]<\epsilon.$$

2.1.2 The *Gap Hamming Distance* Problem

Let $n\in\mathbb{N}$, and let $\zeta_0,\zeta_1>0$. We define the *Gap Hamming Distance* problem as follows:

Definition 2.2. *The* Gap Hamming Distance *problem is the communication complexity problem of computing the partial Boolean function* $\mathsf{GHD}_{n,\zeta_0,\zeta_1}:\{-1,1\}^n\times\{-1,1\}^n\to\{0,1\}$ *given by*

$$\mathsf{GHD}_{n,\zeta_0,\zeta_1}(x,y)=\begin{cases}1 & if \quad \langle x,y\rangle>\zeta_1\\ 0 & if \quad \langle x,y\rangle<-\zeta_0\end{cases}$$

Denote $\mathsf{GHD}=\mathsf{GHD}_{n,\sqrt{n},\sqrt{n}}$.

2.2 Streaming Complexity

Let $\epsilon\geq 0$, $\delta>0$. Let $m,n\in\mathbb{N}$. A *data stream* $\sigma=(a_1,\ldots,a_m)$ is a sequence of elements, each from $[n]=\{1,\ldots,n\}$. We say that the *length* of the stream is m, and the *alphabet size* is n.

A *streaming algorithm* is a space-bounded probabilistic algorithm that gets an element-by-element access to a *data stream*. After each element arrives, the algorithm can no longer access the elements that precede it. At the end of its run, the *streaming algorithm* is required to output (with high probability) a certain function of the *data stream* that it read. When dealing with *streaming algorithms*, the main resource we are concerned with is the size of the space that the algorithm uses.

Formally, a *data stream problem* $\mathcal{P}$ is a collection of functions $\{f_{m,n}:[n]^m\to\mathbb{R}\}_{m,n\in\mathbb{N}}$. That is, a function for every combination of *length* and *alphabet size* of a *data stream*. However, slightly abusing notation for the sake of brevity, we will define each *data stream problem* by a single function (which in fact depends on the length m and alphabet size n of the *data stream*). A δ-error, ϵ-approximation data stream algorithm $\mathcal{A}_{\epsilon,\delta}$ for $\mathcal{P}$ is a probabilistic algorithm that gets a sequential, one pass access to a *data stream* $\sigma=(a_1,\ldots,a_m)$ (where each a_i is a member of $[n]$), and satisfies:

$$\Pr\left[\left|\frac{\mathcal{A}_{\epsilon,\delta}(\sigma)}{f_{m,n}(\sigma)}-1\right|>\epsilon\right]<\delta.$$

If $\epsilon=0$ we say that the *streaming algorithm* is exact.

Last, given a data stream problem $\mathcal{P} = \{f_{m,n} : [n]^m \to \mathbb{R}\}_{m,n\in\mathbb{N}}$ and a data stream $\sigma = (a_1, \ldots, a_m)$ (with alphabet $[n]$) we denote by $\mathcal{P}(\sigma)$ the output of $f_{m,n}(\sigma)$, for the $f_{m,n} \in \mathcal{P}$ that matches the length and alphabet size of σ. Similarly, when applying a family of functions to σ, we in fact apply a specific function in the family, according to the parameters m, n of σ.

2.2.1 The *Distinct Elements* Problem

The *Distinct Elements* problem is the problem of computing the exact number of distinct elements that appear in a data stream, denoted by $F_0(\sigma)$. Formally, we define:

Definition 2.3. *The* Distinct Elements *problem is the data stream problem of computing the exact number of distinct elements in a data stream $\sigma = (a_1, \ldots, a_m)$ (where $a_i \in [n]$ for every i), i.e., computing (exactly):*

$$F_0(\sigma) = \Big| \{i \in \mathbb{N} \ : \ \exists j \in [m] \ \ a_j = i\} \Big|.$$

Note that if we define $0^0 = 0$ then this is exactly the 0'th frequency moment of the stream. Hence the notation F_0.

3 Streaming Algorithms with Probabilistic Proof Systems

In this section we extend the data stream computational model in order to support two types of probabilistic proof systems: $\mathcal{MA}$ algorithms, wherein the streaming algorithm gets a proof that it probabilistically verifies, and $\mathcal{AM}$ algorithms that extend $\mathcal{MA}$ algorithms by adding shared randomness. We study both of these probabilistic proof systems in two variations: in the first, the proof is also being streamed to the verifier, and in the second, the verifier has a free access to the proof. Formal definitions follow.

3.1 MA Streaming Algorithms

Similarly to the way $\mathcal{MA}$ communication complexity protocols are defined, in $\mathcal{MA}$ streaming algorithms we have an omniscient prover (Merlin) who sends a proof to a verifier (Arthur), which is in fact a streaming algorithm that gets both the input stream and the proof (either by a free access or by a one-pass, sequential access). The streaming algorithm computes a function of the input stream. Using the proof we hope to achieve a better space complexity than what the regular streaming model allows.

We start with $\mathcal{MA}$ proofs wherein the proof is being streamed to the verifier. Formally, we define

Definition 3.1. *Let $\epsilon \geq 0$, $\delta > 0$, and let $\mathcal{P} = \{f_{m,n} : [n]^m \to \mathbb{R}\}_{m,n\in\mathbb{N}}$ be a data stream problem. An $\mathcal{MA}$ streaming algorithm for $\mathcal{P}$ is a probabilistic data stream algorithm $\mathcal{A}$, which simultaneously gets two streams: an input stream $\sigma = (a_1, \ldots, a_m)$ (where $a_i \in [n]$ for every*

i) and a proof stream ω; to both it has a sequential, one pass access. Given two functions $S, W : \mathbb{N}^2 \to \mathbb{N}$, we say that an $\mathcal{MA}$ streaming algorithm is $\mathcal{MA}_{\epsilon,\delta}\big(S(m,n), W(m,n)\big)$ if it uses at most $S(m,n)$ bits of space, and satisfies:

1. ***Completeness**: for every $\sigma = (a_1, \ldots, a_m)$ (with* alphabet $[n]$*) there exists a non empty set $\mathcal{W}_\sigma$ of proof streams of length at most $W(m,n)$, such that for every $\omega \in \mathcal{W}_\sigma$ we have,*

$$\Pr\left[\left|\frac{\mathcal{A}(\sigma,\omega)}{f_{m,n}(\sigma)} - 1\right| \leq \epsilon\right] > 1 - \delta$$

2. ***Soundness**: for every $\sigma = (a_1, \ldots, a_m)$ (with* alphabet $[n]$*), and for every $\omega \notin \mathcal{W}_\sigma$ we have*

$$\Pr[\mathcal{A}(\sigma,\omega) \neq \perp] < \delta$$

where $\perp \notin \mathbb{R}$ is a symbol that represents that the algorithm could not verify the correctness of the proof.

The second natural way to define an $\mathcal{MA}$ probabilistic proof system for the data stream model, is by allowing the algorithm a free access to the proof. This leads to the following definition:

Definition 3.2. *Let $\epsilon \geq 0$, $\delta > 0$, and let $\mathcal{P} = \{f_{m,n} : [n]^m \to \mathbb{R}\}_{m,n\in\mathbb{N}}$ be a data stream problem. An $\widehat{\mathcal{MA}}$ streaming algorithm for $\mathcal{P}$ is a probabilistic data stream algorithm $\mathcal{A}^w$, which has a free oracle access to a proof string w. The algorithm gets a stream $\sigma = (a_1, \ldots, a_m)$ (where $a_i \in [n]$ for every i) as an input, to which it has a sequential, one pass access. Given two functions $S, W : \mathbb{N}^2 \to \mathbb{N}$, we say that an $\widehat{\mathcal{MA}}$ streaming algorithm is $\widehat{\mathcal{MA}}_{\epsilon,\delta}\big(S(m,n), W(m,n)\big)$ if it uses at most $S(m,n)$ bits of space, and satisfies:*

1. ***Completeness**: for every $\sigma = (a_1, \ldots, a_m)$ (with* alphabet $[n]$*), there exists a non empty set $\mathcal{W}_\sigma$ of proof strings of length at most $W(m,n)$, such that for every $w \in \mathcal{W}_\sigma$ we have,*

$$\Pr\left[\left|\frac{\mathcal{A}^w(\sigma)}{f_{m,n}(\sigma)} - 1\right| \leq \epsilon\right] > 1 - \delta$$

2. ***Soundness**: for every $\sigma = (a_1, \ldots, a_m)$ (with* alphabet $[n]$*), and for every $w \notin \mathcal{W}_\sigma$ we have*

$$\Pr[\mathcal{A}^w(\sigma) \neq \perp] < \delta$$

where $\perp \notin \mathbb{R}$ is a symbol that represents that the algorithm could not verify the correctness of the proof.

Note that by definition, the model of $\mathcal{MA}$ streaming with a free access to the proof is stronger than the model of $\mathcal{MA}$ streaming with a proof stream. Thus when in Section 6 we prove lower bounds on the $\widehat{\mathcal{MA}}$ streaming complexity, it also implies lower bounds on the $\mathcal{MA}$ streaming complexity.

3.2 AM Streaming Algorithms

We can further extend the data stream model to support an $\mathcal{AM}$ probabilistic proof system. Similarly to the case of $\mathcal{MA}$ proofs, an $\mathcal{AM}$ streaming algorithm receives a proof stream and an input stream, to which it has a sequential, one pass access; except that in $\mathcal{AM}$ proof systems the prover and verifier also share a common random string. Formally, we define

Definition 3.3. *Let $\epsilon \geq 0$, $\delta > 0$, and let $\mathcal{P} = \{f_{m,n} : [n]^m \to \mathbb{R}\}_{m,n\in\mathbb{N}}$ be a data stream problem. An $\mathcal{AM}$ streaming algorithm for $\mathcal{P}$ is a probabilistic data stream algorithm $\mathcal{A}^r$ that has an oracle access to a common random string r, and that is also allowed to make private random coin tosses. The algorithm simultaneously gets two streams: an input stream $\sigma = (a_1, \ldots, a_m)$ (where $a_i \in [n]$ for every i) and a proof stream ω, to both it has a sequential, one pass access. Given two functions $S, W : \mathbb{N}^2 \to \mathbb{N}$, we say that an $\mathcal{AM}$ streaming algorithm is $\mathcal{AM}_{\epsilon,\delta}\big(S(m,n), W(m,n)\big)$ if it uses at most $S(m,n)$ bits of space, and satisfies that for every $\sigma = (a_1, \ldots, a_m)$ (over* alphabet $[n]$*), with probability at least $1 - \delta/2$ (over r) there exists a non empty set $\mathcal{W}_\sigma(r)$ of proof streams of length at most $W(m,n)$, such that:*

1. ***Completeness**: For every $\omega \in \mathcal{W}_\sigma(r)$*

$$\Pr\left[\left|\frac{\mathcal{A}^r(\sigma,\omega)}{f_{m,n}(\sigma)} - 1\right| \leq \epsilon\right] > 1 - \frac{\delta}{2},$$

   where the probability is taken over the private random coin tosses of $\mathcal{A}^r$.

2. ***Soundness**: For $\omega \notin \mathcal{W}_\sigma(r)$*

$$\Pr\left[\mathcal{A}^r(\sigma,\omega) = \perp\right] > 1 - \frac{\delta}{2},$$

   where the probability is taken over the private random coin tosses of $\mathcal{A}^r$, and $\perp \notin \mathbb{R}$ is a symbol that represents that the algorithm could not verify the correctness of the proof.

The randomness complexity *of the algorithm is the total size of the common random string r, and the number of private random coin tosses that the algorithms performs.*

Note that we slightly deviate from the standard definition of an $\mathcal{AM}$ algorithm, by allowing $\mathcal{A}$ to be a probabilistic algorithm with a private random string.

Just as with the $\mathcal{MA}$ streaming model, we can define $\widehat{\mathcal{AM}}$ streaming algorithms by allowing a free access to the proof. Again, by definition the model of $\mathcal{AM}$ streaming with a free access to the proof is stronger than the model of $\mathcal{AM}$ streaming with a proof stream. Our canonical $\mathcal{AM}$ algorithm works for the weaker model, wherein the proof is being streamed, thus our $\mathcal{AM}$ upper bounds also implies $\widehat{\mathcal{AM}}$ upper bounds.

Note 1: In both of the models ($\mathcal{MA}$ and $\mathcal{AM}$), as traditionally done in Arthur-Merlin probabilistic proof systems, we will sometimes describe the $\mathcal{MA}/\mathcal{AM}$ algorithm as an interaction between an omniscient prover Merlin, who sends an alleged proof of a statement

to Arthur, a computationally limited verifier (in our case, a streaming algorithm), who in turn probabilistically verifies the correctness of Merlin's proof.

Note 2: In all of our ($\mathcal{MA}$ and $\mathcal{AM}$) algorithms, we assume without loss of generality that Arthur knows both the length m and the alphabet size n. This can be done since we can insert m, n at the beginning of the proof. Then, Arthur only needs to verify that the length of the stream was indeed m, and that no element was bigger than n. Since all of the algorithms we present in this paper are $\Omega(\log m + \log n)$ in both proof size and space complexity, this does not change their overall asymptotical complexity.

4 The Canonical AM Streaming Algorithm

In this section we show our canonical $\mathcal{AM}$ algorithm. Recall that given a data stream $\sigma = (a_1, \ldots, a_m)$ (over alphabet $[n]$), the *element indicator* $\chi_i : [n] \to \{0,1\}$ of the i'th element ($i \in [m]$) of the stream σ, is the function that indicates whether a given element is in position $i \in [m]$ of σ, i.e., $\chi_i(j) = 1$ if and only if $a_i = j$. Furthermore, let $\chi : [n] \to \{0,1\}^m$ be the *element indicator* of σ, defined by

$$\chi(j) = \big(\chi_1(j), \ldots, \chi_m(j)\big).$$

In addition, given $n \in \mathbb{N}$ we define a *clause* over n variables $x_1, \ldots, x_n$ as a function $C : \{0,1\}^n \to \{0,1\}$ of the form $(y_1 \vee y_2 \vee \ldots \vee y_n)$, where for every $i \in [n]$ the literal y_i is either a variable (x_j), a negation of a variable ($\neg x_j$), or one of the constants $\{0,1\}$.

We prove the following theorem:

Theorem 4.1. *Let $0 \leq \epsilon < 1/2$. Let $\mathcal{P}$ be a data stream problem such that for every $m, n \in \mathbb{N}$ there exists a set of $k = k(m,n)$ clauses $\{C_t\}_{t\in[k]}$ over m variables, and a function $\psi : \{0,1\}^k \to \mathbb{Z}$, such that for every data stream $\sigma = (a_1, \ldots, a_m)$ with alphabet $[n]$,*

$$(1-\epsilon)\mathcal{P}(\sigma) \leq \sum_{j=1}^{n} \psi\big(C_1 \circ \chi(j), \ldots, C_k \circ \chi(j)\big) \leq (1+\epsilon)\mathcal{P}(\sigma).$$

Moreover, we assume that ψ and $\{C_t\}_{t\in[k]}$ are known to the verifier, and that there exists $B \leq \mathrm{poly}(m,n)$ such that $\psi(x) < B$ for every $x \in \{0,1\}^k$. Then, for every $0 < \delta \leq 1$ and every $s, w \in \mathbb{N}$ such that $s \cdot w \geq n$, there exists an explicit $\mathcal{AM}_{\epsilon,\delta}(S,W)$-streaming algorithm for approximating $\mathcal{P}(\sigma)$; where $S = O\big(sk \cdot \mathrm{polylog}(m,n,\delta^{-1})\big)$, $W = O\big(wk \cdot \mathrm{polylog}(m,n,\delta^{-1})\big)$, and the randomness complexity is $\mathrm{polylog}(m,n,\delta^{-1})$.

Proof. Let $0 \leq \epsilon < 1/2$. Let $\mathcal{P}$ be a data stream problem such that for every $m, n \in \mathbb{N}$ there exists a set of $k = k(m,n)$ clauses $\{C_t\}_{t\in[k]}$ over m variables, and a function $\psi : \{0,1\}^k \to \mathbb{Z}$, such that for every data stream $\sigma = (a_1, \ldots, a_m)$ with alphabet $[n]$,

$$(1-\epsilon)\mathcal{P}(\sigma) \leq \sum_{j=1}^{n} \psi\big(C_1 \circ \chi(j), \ldots, C_k \circ \chi(j)\big) \leq (1+\epsilon)\mathcal{P}(\sigma). \tag{4.1}$$

Assume that ψ and $\{C_t\}_{t\in[k]}$ are known to the verifier, and that there exists $B \leq \text{poly}(m,n)$ such that $\psi(x) < B$ for every $x \in \{0,1\}^k$. Observe that since ψ gets $\{0,1\}$ values as inputs, we can think of ψ as a multilinear polynomial. Assume without loss of generality that $k \leq m$ (otherwise the theorem follows trivially).

Let $0 < \delta \leq 1$ and let $s, w \in \mathbb{N}$ such that $s \cdot w \geq n$ (assume for simplicity and without loss of generality that $s \cdot w = n$ exactly). We show that there exists an explicit $\mathcal{AM}_{\epsilon,\delta}(S,W)$-streaming algorithm for approximating $\mathcal{P}(\sigma)$; where

$$\begin{aligned} S &= O\big(sk \cdot \text{polylog}(m,n,\delta^{-1})\big), \\ W &= O\big(wk \cdot \text{polylog}(m,n,\delta^{-1})\big), \end{aligned}$$

and the randomness complexity is $\text{polylog}(m,n,\delta^{-1})$.

Let $\sigma = (a_1,\ldots,a_m)$ be a data stream with alphabet $[n]$. The first step is representing the middle term of (4.1) as a summation of a low degree polynomial over some domain. Specifically, we represent the element indicators $\{\chi_i\}_{i\in[m]}$ as bivariate polynomials over a finite field.

Let p be a sufficiently large (to be determined later) prime number of order

$$\frac{1}{\delta} \cdot \text{poly}(m,n)$$

such that: $p > 2nB > \mathcal{P}(\sigma)$. Let $\mathcal{D}_s(\mathbb{F}_p)$ be any efficiently enumerable subset, of cardinality s, of the field $\mathbb{F}_p$ (e.g., the lexicographically first elements in some representation of the field $\mathbb{F}_p$). Likewise, let $\mathcal{D}_w(\mathbb{F}_p)$ be any efficiently enumerable subset, of cardinality w, of the field $\mathbb{F}_p$. Note that since $n = w \cdot s$, there exists a one-to-one mapping between the domain $[n]$ and the domain $\mathcal{D}_w(\mathbb{F}_p) \times \mathcal{D}_s(\mathbb{F}_p)$. Fix such (efficiently computable) mapping $\pi : [n] \to \mathcal{D}_w(\mathbb{F}_p) \times \mathcal{D}_s(\mathbb{F}_p)$ (e.g., according to the lexicographic order).

For every $i \in [m]$ we can view $\chi_i : [n] \to \{0,1\}$ as a bivariate polynomial $\tilde{\chi}_i : \mathbb{F}_p^2 \to \mathbb{F}_p$ of degree $w-1$ in the first variable (which we denote by x), and degree $s-1$ in the second variable (which we denote by y), such that for every $j \in [n]$ we have $\tilde{\chi}_i \circ \pi(j) = \chi_i(j)$. If we denote $(\alpha_i, \beta_i) \coloneqq \pi(a_i)$, then the extension $\tilde{\chi}_i : \mathbb{F}_p^2 \to \mathbb{F}_p$ is given explicitly by the Lagrange interpolation polynomial:

$$\tilde{\chi}_i(x,y) = \frac{\displaystyle\prod_{\substack{a\in\mathcal{D}_w(\mathbb{F}_p)\\ a\neq\alpha_i}} (x-a) \prod_{\substack{b\in\mathcal{D}_s(\mathbb{F}_p)\\ b\neq\beta_i}} (y-b)}{\displaystyle\prod_{\substack{a\in\mathcal{D}_w(\mathbb{F}_p)\\ a\neq\alpha_i}} (\alpha_i-a) \prod_{\substack{b\in\mathcal{D}_s(\mathbb{F}_p)\\ b\neq\beta_i}} (\beta_i-b)} \tag{4.2}$$

Note that for every $\xi \in \mathcal{D}_s(\mathbb{F}_p)$, the degree of the univariate polynomial $\tilde{\chi}_i(\cdot,\xi) : \mathbb{F}_p \to \mathbb{F}_p$ is at most $w-1$.

Let $\tilde{\chi} : \mathbb{F}_p^2 \to \mathbb{F}_p^m$ be the polynomial extension of the *element indicator* of σ, defined by

$$\tilde{\chi}(x,y) = \big(\tilde{\chi}_1(x,y),\ldots,\tilde{\chi}_m(x,y)\big).$$

Plugging-in the polynomial extensions of the element indicators to (4.1) yields that

$$\widetilde{\mathcal{P}}(\sigma) \coloneqq \sum_{x \in \mathcal{D}_w(\mathbb{F}_p)} \sum_{y \in \mathcal{D}_s(\mathbb{F}_p)} \psi\big(C_1 \circ \tilde{\chi}(x,y), \ldots, C_k \circ \tilde{\chi}(x,y)\big) \tag{4.3}$$

(where the summation is over $\mathbb{Z}$) approximates $\mathcal{P}(\sigma)$ within a multiplicative factor of $1 \pm \epsilon$. Later, we will give analogous expressions of $\mathcal{P}(\sigma) \pmod q$ for prime numbers $q = O(\log p)$.

Next, we replace each clause in (4.3) with a low degree polynomial (over a small finite field) that approximates it. Towards this end, we show the following lemma (originated in [Raz87, Smo87]):

Lemma 4.2. *Let $\delta' > 0$, let q be a prime number, and let $\{C_t\}_{t \in [k]}$ be a set of k clauses over m variables. Using $\mathrm{polylog}(m, k, \delta'^{-1})$ random coin flips, we can construct a set of polynomials $\{p_t : \mathbb{F}_q^m \to \mathbb{F}_q\}_{t \in [k]}$ of degree $O(q \log^{k}/_{\delta'})$ each, such that for every $x \in \{0,1\}^m$,*

$$\Pr\left[\forall t \in [k] \quad p_t(x) = C_t(x)\right] \geq 1 - \delta'$$

(where the probability is taken over the random coin flips performed during the construction of $\{p_t\}_{t \in [k]}$).

Proof. Consider $\mathcal{C} \coloneqq \{C_t\}_{t \in [k]}$, where for every $t \in [k]$, C_t is a clause over m variables. We approximate each $C_t \in \mathcal{C}$ by a polynomial $p_t : \mathbb{F}_q^m \to \mathbb{F}_q$. Recall that every clause in $\mathcal{C}$ is an m-variate disjunction gate that operates on literals, which are either a variable, or a negation of a variable, or one of the constants $\{0,1\}$.

In order to construct a polynomial approximation of a clause $C_t \in \mathcal{C}$, we first replace each negation gate over a variable x in C_t, with the polynomial $1 - x$. Note that this polynomial computes the negation exactly (i.e., no approximation).

Next, we use the method of [Raz87, Smo87] to approximate the m-variate disjunction gate of C_t, by constructing an approximation polynomial in the following way: let $\mathsf{ECC} : \mathbb{F}_q^m \to \mathbb{F}_q^{100m}$ be a linear error correcting code with relative distance $1/3$. Fix

$$L = O\left(\log k + \log \frac{1}{\delta'}\right),$$

such that

$$\left(\frac{2}{3}\right)^L \leq \frac{\delta'}{k},$$

and choose independently and uniformly at random $\iota_1, \ldots, \iota_L \in [100m]$. We build a low degree polynomial approximation for the Boolean disjunction function. Consider $\eta : \mathbb{F}_q^m \to \mathbb{F}_q$, defined by

$$\eta(z_1, \ldots, z_m) = 1 - \prod_{l=1}^{L} \Big(1 - \big(\mathsf{ECC}(z_1, \ldots, z_m)_{\iota_l}\big)^{q-1}\Big).$$

Since ECC is linear, η is a polynomial of degree $O(L \cdot q)$ in the variables $z_1, \ldots, z_m$. Observe that the linearity and the relative distance of ECC, together with Fermat's little theorem implies that for every $(x_1, \ldots, x_m) \in \{0,1\}^m$,

$$\Pr\left[\eta(x_1, \ldots, x_m) \neq \bigvee_{i=1}^{m} x_i\right] \leq \left(\frac{2}{3}\right)^L \leq \frac{\delta'}{k} \tag{4.4}$$

(where the probability is taken over the random choices of $\iota_1, \ldots, \iota_L \in [100m]$). Note that we use the same polynomial η for all of the clauses in $\mathcal{C}$. Thus, the total number of coin flips that we use is $\mathrm{polylog}(m, k, \delta'^{-1})$. The last step of the construction is defining p_t as the composition of the disjunction polynomial η and the literals in the clause C_t.

Note that applying the approximation procedure that we described above to all of the clauses in $\mathcal{C}$, results with a set of k polynomials $\{p_t : \mathbb{F}_q^m \to \mathbb{F}_q\}_{t\in[k]}$, where for every $t \in [k]$ the degree of p_t is $O(q \log k/\delta')$. We conclude the proof of the lemma by noticing that (4.4) together with a union bound imply that for every $x \in \{0,1\}^m$,

$$\Pr\left[\forall t \in [k] \quad p_t(x) = C_t(x)\right] \geq 1 - \delta'$$

(where the probability is over the random choices of $\iota_1, \ldots, \iota_L \in [100m]$). □

Observe that by applying Lemma 4.2 with $\delta' = \delta$ and p as the prime number, we can represent (4.3) as a summation over a polynomial. However, the degree of this polynomial (which is dominated by p), is too high for our needs. Instead, we approximate (4.3) by $O(\log p)$ low degree polynomials.

We start by introducing the necessary notations. Let $Q = \{q_1, \ldots, q_{\rho(c \log p)}\}$ (where $\rho : \mathbb{N} \to \mathbb{N}$ is the prime counting function) be the set of all prime numbers that are smaller or equal to $c \log p$, where c is a constant such that

$$\prod_{q \in Q} q > p.$$

For every $q \in Q$ denote $\mathbb{H}_q \coloneqq \mathbb{F}_{q^{\lambda_q}}$, where λ_q is the minimum integer that satisfies $q^{\lambda_q} > p$. Since $q = O(\log p)$, and by the minimality of λ_q, we have $|\mathbb{H}_q| < pq = O(p \log p)$. Furthermore,

$$\widetilde{\mathcal{P}}(\sigma) \pmod q = \sum_{x \in \mathcal{D}_w(\mathbb{F}_p)} \sum_{y \in \mathcal{D}_s(\mathbb{F}_p)} \psi\big(C_1 \circ \tilde{\chi}(x,y), \ldots, C_k \circ \tilde{\chi}(x,y)\big) \pmod q \tag{4.5}$$

(where we can think of the summation over $\mathbb{Z}$ modulo q, as summation over $\mathbb{F}_q$). Denote

$$\widetilde{\mathcal{P}}_q(\sigma) \coloneqq \widetilde{\mathcal{P}}(\sigma) \pmod q.$$

Analogously to the definitions for $\mathbb{F}_p$; for every prime $q \in Q$ we define efficiently enumerable subsets $\mathcal{D}_s(\mathbb{H}_q)$, $\mathcal{D}_w(\mathbb{H}_q)$ of $\mathbb{H}_q$, with cardinality s, w (respectively), and a one-to-one mapping $\pi_q : [n] \to \mathcal{D}_w(\mathbb{H}_q) \times \mathcal{D}_s(\mathbb{H}_q)$. For every $i \in [m]$, we can view $\chi_i : [n] \to \{0,1\}$ as

a bivariate polynomial $\tilde{\chi}_i^q : \mathbb{H}_q^2 \to \mathbb{H}_q$ of degree $w-1$ in the first variable (which we denote by x), and degree $s-1$ in the second variable (which we denote by y), such that for every $j \in [n]$ we have $\tilde{\chi}_i^q \circ \pi_q(j) = \chi_i(j)$. Let $\tilde{\chi}^q : \mathbb{H}_q^2 \to \mathbb{H}_q^m$ be defined by

$$\tilde{\chi}^q(x,y) = \big(\tilde{\chi}_1^q(x,y), \ldots, \tilde{\chi}_m^q(x,y)\big).$$

Moreover, we can think of the multilinear polynomial $\psi : \{0,1\}^k \to \mathbb{Z}$ as a multilinear polynomial $\widetilde{\psi} : \mathbb{F}_p^k \to \mathbb{F}_p$ (recall that $\psi(x) < B < p$ for every $x \in \{0,1\}^k$). Let $\widetilde{\psi}_q : \mathbb{F}_q^k \to \mathbb{F}_q$ be the polynomial function defined by the formal polynomial (i.e., a summation of monomials multiplied by coefficients) $\widetilde{\psi}$, where we take each coefficient of $\widetilde{\psi}$ modulo q. Since $\mathbb{F}_q$ is a subfield of $\mathbb{H}_q$, we can also view $\widetilde{\psi}_q$ as a multilinear polynomial from $\mathbb{H}_q^k$ to $\mathbb{H}_q$.

Thus, we can express (4.5) as follows:

$$\widetilde{\mathcal{P}}_q(\sigma) = \sum_{x \in \mathcal{D}_w(\mathbb{H}_q)} \sum_{y \in \mathcal{D}_s(\mathbb{H}_q)} \widetilde{\psi}_q\big(C_1 \circ \tilde{\chi}^q(x,y), \ldots, C_k \circ \tilde{\chi}^q(x,y)\big) \tag{4.6}$$

(where the summation is over $\mathbb{H}_q$, which in this case is equal to summation over $\mathbb{F}_q$, hence the modulo q).[2]

For every $q \in Q$, we apply Lemma 4.2 with $\delta' = \frac{\delta}{2nc\log p}$, and q as the prime number. We get a set of polynomials

$$\left\{p_t : \mathbb{F}_q^m \to \mathbb{F}_q\right\}_{t \in [k]}$$

(for every $q \in Q$), of degree $O\left(q \log \frac{kn \log p}{\delta}\right)$ each, such that for every $x \in \{0,1\}^m$,

$$\Pr\left[\forall t \in [k] \quad p_t(x) = C_t(x)\right] \geq 1 - \frac{\delta}{2nc\log p} \tag{4.7}$$

(where the probability is taken over the random coin flips performed during the construction of $\{p_t\}_{t\in[k]}$).

Since $\mathbb{F}_q$ is a subfield of $\mathbb{H}_q$, we can view $p_t : \mathbb{F}_q^m \to \mathbb{F}_q$ as a polynomial $\widetilde{p}_t : \mathbb{H}_q^m \to \mathbb{H}_q$ (for every $t \in [k]$). Then, for every $x \in \mathbb{F}_q^m$ we have $\widetilde{p}_t(x) = p_t(x)$. Thus, we get the following set of polynomials:

$$\left\{\widetilde{p}_t : \mathbb{H}_q^m \to \mathbb{H}_q\right\}_{t \in [k]},$$

where for every $t \in [k]$, the degree of $\widetilde{p}_t$ is $O\left(q \log \frac{kn \log p}{\delta}\right)$.

Applying a union bound, and using (4.7) yields:

$$\Pr\left[\widetilde{\mathcal{P}}_q(\sigma) = \sum_{x \in \mathcal{D}_w(\mathbb{H}_q)} \sum_{y \in \mathcal{D}_s(\mathbb{H}_q)} \widetilde{\psi}_q\big(\widetilde{p}_1 \circ \tilde{\chi}^q(x,y), \ldots, \widetilde{p}_k \circ \tilde{\chi}^q(x,y)\big)\right] \geq 1 - \frac{\delta}{2c\log p} \tag{4.8}$$

[2]Since for every $x \in \mathcal{D}_w(\mathbb{H}_q)$ and $y \in \mathcal{D}_s(\mathbb{H}_q)$ we have $\big(C_1 \circ \tilde{\chi}^q(x,y), \ldots, C_k \circ \tilde{\chi}^q(x,y)\big) \in \{0,1\}^k$, then each summand is in $\mathbb{F}_q$. Hence we can think of the summation as summation over $\mathbb{F}_q$.

(where the probability is taken over the random coin flips performed during the construction of $\{p_t\}_{t\in[k]}$, and the summation is over $\mathbb{H}_q$).[3]

Next, we define the polynomial $\omega_q : \mathbb{H}_q \to \mathbb{H}_q$ by

$$\omega_q(x) = \sum_{y\in\mathcal{D}_s(\mathbb{H}_q)} \widetilde{\psi}_q\big(\widetilde{p_1}\circ\tilde{\chi}^q(x,y),\ldots,\widetilde{p_k}\circ\tilde{\chi}^q(x,y)\big)$$

(where the summation is over $\mathbb{H}_q$). Note that for every $t\in[k]$, the composition of $\widetilde{p_t}$ and $\tilde{\chi}^q$ is a polynomial of degree

$$O\left(wq\log\frac{kn\log p}{\delta}\right)$$

in x (the first variable). Hence, by the multilinearity of $\widetilde{\psi}$,

$$\deg(\omega_q) = O\left(wkq\log\frac{kn\log p}{\delta}\right). \tag{4.9}$$

By (4.8) we have,

$$\Pr\left[\widetilde{\mathcal{P}}_q(\sigma) = \sum_{x\in\mathcal{D}_w(\mathbb{H}_q)}\omega_q(x)\right] \geq 1-\frac{\delta}{2c\log p} \tag{4.10}$$

(where the probability is taken over the random coin flips performed during the construction of $\{p_t\}_{t\in[k]}$, and the summation is over $\mathbb{H}_q$).

Once we established the above, we can finally describe Merlin's proof stream. The proof stream φ consists of all the proof polynomials $\{\omega_q\}_{q\in Q}$. We send each polynomial by its list of coefficients, thus we need at most

$$O\left(|Q|wk\log(p)\log\left(\frac{kn\log p}{\delta}\right)\cdot\log(p\log p)\right)$$

bits in order to write down the proof stream. Since $|Q| < c\log p$, we conclude:

Claim 4.3. *the total size of Merlin's proof stream φ is*

$$O\Big(wk\cdot\operatorname{polylog}\left(m,n,\delta^{-1}\right)\Big).$$

Observe that it is possible to reconstruct $\widetilde{P}(\sigma)$ from the polynomials given in Merlin's proof. We formalize this claim as follows:

Claim 4.4. *Given the set of values $\{\sum_{x\in\mathcal{D}_w(\mathbb{H}_q)}\omega_q(x)\}_{q\in Q}$, it is possible to compute $\widetilde{P}(\sigma)$ with probability $1-\delta/2$ (over the random coin tosses that were performed during the construction of $\{\omega_q\}_{q\in Q}$).*

[3]Again, since for every $x\in\mathcal{D}_w(\mathbb{H}_q)$ and $y\in\mathcal{D}_s(\mathbb{H}_q)$ we have $\big(\widetilde{p_1}\circ\tilde{\chi}^q(x,y),\ldots,\widetilde{p_k}\circ\tilde{\chi}^q(x,y)\big)\in\{0,1\}^k$, then each summand is in $\mathbb{F}_q$. Hence, the summation is modulo q.

Proof. Note that for every $q \in Q$ we have

$$\Pr\left[\sum_{x\in\mathcal{D}_w(\mathbb{H}_q)} \omega_q(x) = \widetilde{\mathcal{P}}(\sigma) \pmod q\right] \geq 1 - \frac{\delta}{2c\log p}.$$

Hence,

$$\Pr\left[\forall q \in Q \quad \sum_{x\in\mathcal{D}_w(\mathbb{H}_q)} \omega_q(x) = \widetilde{\mathcal{P}}(\sigma) \pmod q\right] \geq 1 - \frac{\delta}{2}.$$

By the Chinese remainder theorem, given $\{\widetilde{\mathcal{P}}(\sigma) \pmod q\}_{q\in Q}$ we can calculate

$$\widetilde{\mathcal{P}}(\sigma) \pmod{\prod_{q\in Q} q}.$$

Since we've chosen Q such that $\prod_{q\in Q} q > p$, the claim follows. □

Another important property of the polynomials $\{\omega_q\}_{q\in Q}$ in the proof stream, is that given a sequential, one-pass access to the input stream, it is possible to efficiently evaluate each polynomial at a specific point. Formally, we show:

Lemma 4.5. *For every $q \in Q$, there exists a streaming algorithm $\mathcal{A}_q$ with an access to the common random string r, such that given a point in the finite field $\xi \in \mathbb{H}_q$, and a sequential, one-pass access to the input stream σ, the streaming algorithm $\mathcal{A}_q$ can evaluate $\omega_q(\xi)$ using $O\big(sk \cdot \mathrm{polylog}(m, n, \delta^{-1})\big)$ bits of space.*

Proof. First, recall that the descriptions of $\{C_t\}_{t\in[k]}$ and ψ are known to the verifier. Note that in order to compute $\omega_q(\xi)$ it is sufficient to compute and store the values of

$$\left\{\widetilde{p_t}\big(\tilde{\chi}_1^q(\xi, y), \ldots, \tilde{\chi}_m^q(\xi, y)\big)\right\}_{t\in[k], y\in\mathcal{D}_s(\mathbb{H}_q)},$$

where $\{\widetilde{p_t}\}_{t\in[k]}$ are the approximation polynomials of the clauses $\{C_t\}_{t\in[k]}$ over $\mathbb{H}_q$. Given these values we can compute

$$\left\{\widetilde{\psi}_q\Big(\widetilde{p_1}\big(\tilde{\chi}_1^q(\xi, y), \ldots, \tilde{\chi}_m^q(\xi, y)\big), \ldots, \widetilde{p_k}\big(\tilde{\chi}_1^q(\xi, y), \ldots, \tilde{\chi}_m^q(\xi, y)\big)\Big)\right\}_{y\in\mathcal{D}_s(\mathbb{H}_q)}$$

monomial-by-monomial according to the description of ψ, and then compute $\omega_q(\xi)$ by summing term-by-term.

Before we describe the algorithm, recall that during the construction of $\{p_t\}_{t\in[k]}$ we defined an error correcting code $\mathsf{ECC} : \mathbb{F}_q^m \to \mathbb{F}_q^{100m}$ with relative distance $1/3$. Note that since ECC is a linear function, we can extend it (via the linear extension) to $\mathbb{H}_q$. We fixed

$$L = O\left(\log k + \log\frac{1}{\delta'}\right) = O\left(\log k + \log\frac{n\log p}{\delta}\right),$$

and chose independently and uniformly $\iota_1, \ldots, \iota_L \in [100m]$, using the common random string r. Finally we approximated each of the $\vee$ gates by the following polynomial,

$$\eta(z_1, \ldots, z_m) = 1 - \prod_{l=1}^{L} \Big(1 - (\mathsf{ECC}(z_1, \ldots, z_m)_{\iota_l})^{q-1}\Big). \tag{4.11}$$

Note that in order to compute

$$\widetilde{p}_t\big(\tilde{\chi}_1^q(\xi, y), \ldots, \tilde{\chi}_m^1(\xi, y)\big)$$

for all $t \in [k]$ and $y \in \mathcal{D}_s(\mathbb{H}_q)$, it is sufficient to compute

$$\mathsf{ECC}(\ell_1^t(\xi, y), \ldots, \ell_m^t(\xi, y))_{\iota_l}$$

(where for every $i \in [m]$ and $t \in [k]$ the value $\ell_i^t(\xi, y)$ is either $\tilde{\chi}_i^q(\xi, y)$, or $1 - \tilde{\chi}_i^q(\xi, y)$, or one of the constants $\{0, 1\}$; depending on the clause C_t), for all $\iota_l \in \{\iota_1, \ldots, \iota_L\}$, $t \in [k]$, and $y \in \mathcal{D}_s(\mathbb{H}_q)$. Then we can compute $\widetilde{p}_t\big(\tilde{\chi}_1^q(\xi, y), \ldots, \tilde{\chi}_m^q(\xi, y)\big)$ according to (4.11).

Since ECC is a linear error correcting code, we can compute each

$$\mathsf{ECC}(\ell_1^t(\xi, y), \ldots, \ell_m^t(\xi, y))_{\iota_l}$$

incrementally. That is, we read the data stream σ element-by-element. At each step, when the i'th element arrives ($i \in [m]$), for every $y \in \mathcal{D}_s(\mathbb{H}_q)$ we compute $\tilde{\chi}_i^q(\xi, y)$ according to (4.2), and then $\ell_i^t(\xi, y)$ according to the description of C_t. By the linearity of ECC we can compute $\mathsf{ECC}(\ell_1^t(\xi, y), \ldots, \ell_m^t(\xi, y))_{\iota_l}$ by incrementally adding each

$$\mathsf{ECC}(0, \ldots, 0, \ell_i^t(\xi, y), 0, \ldots, 0)_{\iota_1}$$

at the i'th step.

Observe that during the run over σ, the entire computation is performed element-by-element, and that we used at most $O\left(|\mathcal{D}_s(\mathbb{H}_q)| \cdot k \cdot L \cdot \log p\right)$ bits of space. Thus the overall space complexity is

$$O\Big(sk \cdot \operatorname{polylog}\left(m, n, \delta^{-1}\right)\Big).$$

□

The last lemma helps us to show that with high probability Merlin cannot cheat Arthur by using maliciously chosen proof polynomials. We show that by evaluating the actual proof polynomials at a randomly chosen point, Arthur can detect a false proof with high probability. Formally:

Lemma 4.6. *For every $q \in Q$, given a polynomial $\hat{\omega}_q : \mathbb{H}_q \to \mathbb{H}_q$ of degree at most $O\left(wkq \log \frac{kn \log p}{\delta}\right)$,[4] if $\hat{\omega}_q \neq \omega_q$ then:*

$$\Pr[\hat{\omega}_q(\xi) = \omega_q(\xi)] \leq \frac{\delta}{2},$$

where the probability is taken over uniformly choosing at random an element $\xi \in \mathbb{H}_q$.

[4]More precisely, the degree is exactly as in 4.9.

Proof. Let ξ be an element uniformly chosen from $\mathbb{H}_q$. By the Schwartz-Zippel Lemma, we have

$$\Pr[\hat{\omega}_q(\xi) = \omega_q(\xi)] \leq \frac{\max\{\deg(\omega_q), \deg(\hat{\omega}_q)\}}{|\mathbb{H}_q|} \leq \frac{\delta}{2},$$

where in order to get the last inequality we fix p to be a sufficiently large prime number, of order

$$\frac{1}{\delta} \cdot \mathrm{poly}(m, n).$$

□

Finally, building upon the aforementioned lemmas, we can present the $\mathcal{AM}$ algorithm for the approximation of $\mathcal{P}(\sigma)$:

The prover (Merlin):

1. Choose $\iota_1, \ldots, \iota_L \in [100m]$ using the common random string r.
2. Construct φ that consists of all the proof polynomials $\{\omega_q\}_{q \in Q}$.
3. Send (via streaming) $\varphi = \{\omega_q\}_{q \in Q}$ to the verifier.

The verifier (Arthur):

1. For every $q \in Q$, select uniformly at random $\xi_q \in \mathbb{H}_q$ (where the selection uses Arthur's private random coin tosses).
2. Read Merlin's proof stream $\varphi = \{\hat{\omega}_q\}_{q \in Q}$ and (incrementally) compute:
   (a) $\{\hat{\omega}_q(\xi_q)\}_{q \in Q}$.
   (b) $\left\{\sum_{x \in \mathcal{D}_w(\mathbb{H}_q)} \hat{\omega}_q(x)\right\}_{q \in Q}$.
3. Run $\{\mathcal{A}_q\}_{q \in Q}$ in parallel, in order to compute $\{\omega_q(\xi_q)\}_{q \in Q}$.
4. If there exists $q \in Q$ for which $\omega_q(\xi_q) \neq \hat{\omega}_q(\xi_q)$, return $\perp$.
5. Otherwise, use $\left\{\sum_{x \in \mathcal{D}_w(\mathbb{H}_q)} \hat{\omega}_q(x)\right\}_{q \in Q}$ to extract and return $\widetilde{\mathcal{P}}(\sigma)$.

Figure 1: The Canonical $\mathcal{AM}$ streaming algorithm

Last, we show that the aforementioned algorithm is an $\mathcal{AM}_{\epsilon,\delta}(S, W)$-streaming algorithm for $\mathcal{P}(\sigma)$, where

- $S = O\big(sk \cdot \text{polylog}(m, n, \delta^{-1})\big)$,
- $W = O\big(wk \cdot \text{polylog}(m, n, \delta^{-1})\big)$.

Indeed, given $\epsilon \geq 0$, $\delta > 0$, a common random string r, and a data stream problem $\mathcal{P}$, our algorithm is a probabilistic data stream algorithm (denote it by $\mathcal{A}$), which has an oracle access to r. The algorithm simultaneously gets two streams: an input stream σ and a proof stream φ, to both it has a sequential, one pass access. According to Claim 4.3:

$$W = O\big(wk \cdot \text{polylog}(m, n, \delta^{-1})\big).$$

As for the space complexity of $\mathcal{A}$, note that $\mathcal{A}$ stores $O(\log p)$ random values $\{\xi_q\}_{q \in Q}$ of size $O(\log p)$ each, which takes $\text{polylog}(m, n, \delta^{-1}))$ bits of space. In addition it uses $\text{polylog}(m, n, \delta^{-1}))$ bits of space for computing

1. $\{\hat{\omega}_q(\xi_q)\}_{q \in Q}$.
2. $\left\{\sum_{x \in \mathcal{D}_w(\mathbb{H}_q)} \hat{\omega}_q(x)\right\}_{q \in Q}$.

Observe that these values can be computed incrementally using a sequential, one-pass access to φ, simply by evaluating the polynomials monomial-by-monomial. According to Lemma 4.5, each of the $O(\log p)$ algorithms $\{\mathcal{A}_q\}_{q\in Q}$ we run in parallel takes

$$O\big(sk \cdot \text{polylog}(m, n, \delta^{-1})\big)$$

bits of space. Thus the total space complexity is $S = O\big(sk \cdot \text{polylog}(m, n, \delta^{-1})\big)$.

Recall that the only time that the algorithm used the common random string r, is while building the approximation polynomial for the disjunction in each $\{\omega_q\}_{q\in Q}$. Since we constructed $|Q|$ such polynomials, and by Lemma 4.2, the total number of random bits we read from r is $\text{polylog}(m, n, \delta^{-1})$. Furthermore, $\mathcal{A}$ also uses only $\text{polylog}(m, n, \delta^{-1})$ private random coin tosses, as the only randomness it needs is for the selection of random $\xi_q \in \mathbb{H}_q$ for every $q \in Q$. Thus, the total randomness complexity of the algorithm is $\text{polylog}(m, n, \delta^{-1})$.

We finish the proof by showing the correctness of the algorithm:

1. **Completeness:** Assuming Merlin is honest, i.e., $\omega_q = \hat{\omega}_q$ for every $q \in Q$; then by Claim 4.4 we can calculate $\widetilde{\mathcal{P}}(\sigma)$ with probability $1 - \delta/2$ over the common random string r, and by (4.3) we have

   $$(1-\epsilon)\mathcal{P}(\sigma) \leq \widetilde{\mathcal{P}}(\sigma) \leq (1+\epsilon)\mathcal{P}(\sigma).$$

   Hence:

   $$\Pr\left[\left|\frac{\mathcal{A}(\sigma,\varphi)}{\mathcal{P}(\sigma)} - 1\right| \leq \epsilon\right] \geq 1 - \frac{\delta}{2}$$

2. **Soundness**: If Merlin is dishonest, i.e., there exists $q \in Q$ for which $\omega_q \neq \hat{\omega}_q$, then by Lemma 4.6,

   $$\Pr[\mathcal{A}(\sigma,\varphi) \neq \perp] \leq \frac{\delta}{2},$$

   where the probability is taken over the private random coin tosses that $\mathcal{A}$ performs.

□

5 The MA Communication Complexity of *Gap Hamming Distance*

In this section we show that every $\mathcal{MA}$ communication complexity protocol for the *Gap Hamming Distance* problem (GHD) that communicates T bits and uses a proof of length W, must satisfy $T \cdot W = \Omega(n)$, and therefore $T + W = \Omega(\sqrt{n})$.

In Section 6, we will use the lower bound on the $\mathcal{MA}$ communication complexity of GHD to show a lower bound on the $\widehat{\mathcal{MA}}$ streaming complexity of the *Distinct Elements* problem. We note that the lower bound on the $\mathcal{MA}$ communication complexity of GHD also implies

a lower bound on the $\widehat{\mathcal{MA}}$ streaming complexity of computing the empirical entropy of a data stream (see [CBM06] for a formal definition of the *Empirical Entropy* problem).

For completeness, we show an $\mathcal{MA}$ communication complexity protocol for GHD that communicates $O(T \log n)$ bits and uses a proof of length $O(W \log n)$, for every $T \cdot W \geq n$. Thus we have a tight bound (up to logarithmic factors) of $T \cdot W = \tilde{\Omega}(n)$.

5.1 Lower bound

In order to prove our lower bound on the $\mathcal{MA}$ communication complexity of *Gap Hamming Distance*, we first show a lower bound on the $\mathcal{MA}$ communication complexity of *Gap Orthogonality*, a problem wherein each party gets a vector in $\{-1,1\}^n$ and needs to tell whether the vectors are nearly orthogonal, or far from being orthogonal. We then apply the reduction from the *Gap Orthogonality* problem to the *Gap Hamming Distance* problem (following [She11]), and obtain our lower bound.

Formally, the *Gap Orthogonality* problem is defined as follows:

Definition 5.1. *Let n be an integer, and let $\zeta_0, \zeta_1 > 0$. The* Gap Orthogonality *problem is the communication complexity problem of computing the partial Boolean function $\mathsf{ORT}_{n,\zeta_0,\zeta_1} : \{-1,1\}^n \times \{-1,1\}^n \to \{0,1\}$ given by*

$$\mathsf{ORT}_{n,\zeta_0,\zeta_1}(x,y) = \begin{cases} 1 & if \quad |\langle x,y\rangle| < \zeta_1 \\ 0 & if \quad |\langle x,y\rangle| > \zeta_0 \end{cases}.$$

Denote $\mathsf{ORT} = \mathsf{ORT}_{n,\frac{\sqrt{n}}{4},\frac{\sqrt{n}}{8}}$.

We restate the following theorem from [She11], which given two finite sets X, Y, guaranties that if the inner product of a random vector from X and a random vector from Y is highly concentrated around 0, then $X \times Y$ must be a small rectangle.

Theorem 5.2. *Let $\delta > 0$ be a sufficiently small constant, and let $X, Y \subseteq \{-1,1\}^n$ be two sets, such that*

$$\Pr\left[|\langle x,y\rangle| > \frac{\sqrt{n}}{4}\right] < \delta$$

(where the probability is taken over selecting independently and uniformly at random $x \in X$ and $y \in Y$), then

$$4^{-n}|X||Y| = e^{-\Omega(n)}.$$

Denote the uniform distribution on $\{-1,1\}^n \times \{-1,1\}^n$ by μ. We get the next immediate corollary of Theorem 5.2,

Corollary 5.3. *There exists a (sufficiently small) constant $\delta > 0$ such that for every rectangle $R \subseteq \{-1,1\}^n \times \{-1,1\}^n$ with $\mu(R) > 2^{-\delta n}$ we have*

$$\mu\left(R \cap \mathsf{ORT}^{-1}(0)\right) \geq \delta\mu(R).$$

Proof. Assume by contradiction that there exists a rectangle $R \coloneqq X \times Y \subseteq \{-1,1\}^n \times \{-1,1\}^n$ with $\mu(R) > 2^{-\delta n}$ that satisfies

$$\mu\left(R \cap \mathsf{ORT}^{-1}(0)\right) < \delta\mu(R). \tag{5.1}$$

Observe that

$$\Pr\left[|\langle x,y\rangle| > \frac{\sqrt{n}}{4}\right] = \frac{\mu\left(R \cap \mathsf{ORT}^{-1}(0)\right)}{\mu(R)}$$

(where the probability is taken over selecting independently and uniformly at random $x \in X$ and $y \in Y$). Hence we can write (5.1) as

$$\Pr\left[|\langle x,y\rangle| > \frac{\sqrt{n}}{4}\right] < \delta. \tag{5.2}$$

Note that

$$\mu(R) = \frac{|X|}{2^n} \cdot \frac{|Y|}{2^n} = 4^{-n}|X||Y|.$$

If we choose δ to be sufficiently small, then (5.2) guaranties the precondition of Theorem 5.2, and we get that $\mu(R) = e^{-\Omega(n)}$, in contradiction to the assumption that $\mu(R) > 2^{-\delta n}$. □

In particular, Corollary 5.3 implies that every rectangle $R \subseteq \{-1,1\}^n \times \{-1,1\}^n$ satisfies

$$\mu\left(R \cap \mathsf{ORT}^{-1}(0)\right) \geq \delta\mu(R) - 2^{-\delta n}. \tag{5.3}$$

Next, using well known techniques (cf. [RS04]), we show a lower bound on the $\mathcal{MA}$ communication complexity of ORT, relying on Corollary 5.3. Formally, we prove

Theorem 5.4. *Let ϵ be a positive constant such that $\epsilon < \frac{1}{2}$. For every $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for ORT we have $T \cdot W = \Omega(n)$, hence $T + W = \Omega(\sqrt{n})$.*

Proof. Fix n. Denote $\mathcal{R} = \{-1,1\}^n \times \{-1,1\}^n$. Assume that there exists an $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for ORT; denote it by $\mathcal{P}$. By a simple amplification argument we get that there exists an $\mathcal{MA}_{\epsilon'}(k,W)$ communication complexity protocol for ORT, where $k = O(T \cdot W)$ and $\epsilon' = 2^{-CW}$ (for an arbitrary large constant C); denote it by $\mathcal{P}'$.

Assume by contradiction that $k = o(n)$. We will show that our assumption that k is asymptotically smaller than n implies that the error probability of $\mathcal{P}'$ is greater than 2^{-CW}, in contradiction.

Denote Merlin's proof, a binary string of size at most W bits, by w. Denote the random string that $\mathcal{P}'$ uses by s. Denote by $R_{s,w,h} \subseteq \mathcal{R}$ the set of all input pairs $(x,y) \in \mathcal{R}$ such that the history of (x,y,s,w) is h.[5] We state the following Lemma from [RS04]:

[5]For any input pair $(x,y) \in \mathcal{R}$ and any assignment s to the random string of $\mathcal{P}'$ and any assignment w to the proof supplied to the players, the string of communication bits exchanged by the two players on the inputs (x,y), using the random string s and the proof w, is called the history of (x,y,s,w).

Lemma 5.5. *For every s, w, h we have $R_{s,w,h} = X_{s,w,h} \times Y_{s,w,h}$ (where $X_{s,w,h} \subseteq \{-1,1\}^n$ and $Y_{s,w,h} \subseteq \{-1,1\}^n$), and for every s, w the family $\{R_{s,w,h}\}_{h \in \{0,1\}^k}$ is a partition of $\mathcal{R}$.*

Denote the answer that $\mathcal{P}'$ gives on (x, y, s, w) by $\mathcal{P}'(x, y, s, w)$. Since the answer of $\mathcal{P}'$ on inputs in $R_{s,w,h}$ does not depend on x and y, then for every input pair in $R_{s,w,h}$ the answer $\mathcal{P}'(x, y, s, w)$ is the same; denote it by $\mathcal{P}'(s, w, h)$. Next, define $H_0 \subseteq \mathcal{R}$ to be the set of all input pairs $(x, y) \in \mathcal{R}$ such that

$$|\langle x, y \rangle| > \frac{\sqrt{n}}{4},$$

and define $H_1 \subseteq \mathcal{R}$ to be the set of all input pairs $(x, y) \in \mathcal{R}$ such that

$$|\langle x, y \rangle| < \frac{\sqrt{n}}{8}.$$

Note that if we choose $x = (x_1, \ldots, x_n) \in \{-1,1\}^n$ and $y = (y_1, \ldots, y_n) \in \{-1,1\}^n$ independently and uniformly at random, then for every $i \in [n]$ the product $x_i \cdot y_i$ is also uniformly distributed. Thus, if we choose $z = (z_1, \ldots, z_n) \in \{-1,1\}^n$ uniformly at random, then

$$\mu(H_1) = \Pr_{(x,y) \in \mathcal{R}} \left[|\langle x, y \rangle| < \frac{\sqrt{n}}{8} \right] = \Pr_{z \in \{-1,1\}^n} \left[\left| \sum_{i=1}^{n} z_i \right| < \frac{\sqrt{n}}{8} \right] \geq c, \tag{5.4}$$

for some universal constant c.

Next, for every rectangle $R \subseteq \mathcal{R}$, denote by $\alpha(R)$ the measure of R in $\mathcal{R}$. Denote by $\beta_0(R)$ the measure of $R \cap H_0$ in H_0, and denote by $\beta_1(R)$ the measure of $R \cap H_1$ in H_1. Under these notations, we see that (5.3) implies that there exists a universal constant $\delta > 0$ such that for any rectangle $R \subseteq \mathcal{R}$ we have

$$\beta_0(R) \geq \delta \cdot \alpha(R) - 2^{-\delta n}.$$

According to Equation 5.4, we know that H_1 is a set of probability at least c in $\mathcal{R}$. Hence for every rectangle $R \subseteq \mathcal{R}$ we have $\beta_1(R) \leq 1/c \cdot \alpha(R)$. Therefore we have the following corollary,

Corollary 5.6. *There exist universal constants $\delta, \delta' > 0$ such that every rectangle $R \subseteq \mathcal{R}$ satisfies*

$$\beta_0(R) \geq \delta' \cdot \beta_1(R) - 2^{-\delta n}.$$

For any s, w, denote by $A_0(s, w) \subseteq \mathcal{R}$ the union of all sets $R_{s,w,h}$ such that $\mathcal{P}'(s, w, h) = 0$, and denote by $A_1(s, w) \subseteq \mathcal{R}$ the union of all sets $R_{s,w,h}$ such that $\mathcal{P}'(s, w, h) = 1$. Observe that $A_0(s, w)$ and $A_1(s, w)$ are disjoint, and that $A_0(s, w) \cup A_1(s, w) = \mathcal{R}$.

Since each of $A_0(s, w)$ and $A_1(s, w)$ is a union of at most 2^k of the sets $X_{s,w,h} \times Y_{s,w,h}$, we see that Corollary 5.6 implies

$$\beta_0(A_1(s, w)) \geq \delta' \cdot \beta_1(A_1(s, w)) - 2^k \cdot 2^{-\delta n} \geq \delta' \cdot \beta_1(A_1(s, w)) - o(2^{-W}). \tag{5.5}$$

Recall that $\beta_1(A_1(s,w))$ is the fraction of inputs (x,y) in H_1 such that $\mathcal{P}'(x,y,s,w)=1$, and that H_1 is the set of ones of the problem. Thus for every input (x,y) in H_1 there exists w such that $(x,y)\in A_1(s,w)$ with probability of at least $(1-\epsilon')$ over s. Since the number of possible proofs w is at most 2^W, by an averaging argument we get that there exists a proof that corresponds to at least 2^{-W} fraction of the inputs in H_1. Formally speaking, there exists at least one binary string w of size at most W, and a set $H_1'\subseteq H_1$ that satisfies $\beta_1(H_1')\geq 2^{-W}$, such that that for every $(x,y)\in H_1'$,

$$\Pr_s[(x,y)\in A_1(s,w)]\geq 1-\epsilon'.$$

Therefore, there exists a constant c_0, such that with constant probability (over the random string s),

$$\beta_1(A_1(s,w))>2^{-(W+c_0)}.$$

Hence, by (5.5), with constant probability (over the random string s),

$$\beta_0(A_1(s,w))\geq \delta'\cdot 2^{-W-c_0}-o(2^{-W})\geq c_1\delta'\cdot 2^{-W}.$$

for some constant c_1. However, recall that $\beta_0(A_1(s,w))$ is the fraction of inputs (x,y) in H_0 for which $\mathcal{P}'(x,y,s,w)$ returns 1. Thus there exists a constant c_2 such that,

$$\Pr\left[\mathcal{P}'(x,y,s,w)=1\right]\geq c_2\delta'\cdot 2^{-W}$$

(where the probability is taken over both the random string s, and the uniform selection of $(x,y)\in H_0$). But H_0 is the set of zeros of the problem, so for every $(x,y)\in H_0$ the protocol answers 1 with probability at most $\epsilon'\leq 2^{-CW}$ (for an arbitrary large constant C), which is a contradiction. □

We established that for every $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for ORT we have $T\cdot W=\Omega(n)$. According to the duplication argument in [She11], Theorem 5.4 implies the following corollary for slightly different parameters of the orthogonality problem.

Corollary 5.7. *Let ϵ be a positive constant such that $\epsilon<\frac{1}{2}$. For every $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for $\mathsf{ORT}_{n,2\sqrt{n},\sqrt{n}}(x,y)$ we have $T\cdot W=\Omega(n)$, hence $T+W=\Omega(\sqrt{n})$.*

Next, we state the following reduction from [She11] (repharsed):

Lemma 5.8. *Let $n\in\mathbb{N}$ be a perfect square. For every input $x\in\{-1,1\}^n$ denote by x^m ($m\in\mathbb{N}$) the string of length $n\cdot m$ that is composed of x concatenated to itself $m-1$ times. Then, for every $(x,y)\in\mathsf{ORT}^{-1}_{n,2\sqrt{n},\sqrt{n}}(0)\cup\mathsf{ORT}^{-1}_{n,2\sqrt{n},\sqrt{n}}(1)$ we have*

$$\begin{aligned}\mathsf{ORT}_{n,2\sqrt{n},\sqrt{n}}(x,y) &= \neg\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}\left(x^{10}(-1)^{15\sqrt{n}},y^{10}(+1)^{15\sqrt{n}}\right)\\ &\quad\wedge\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}\left(x^{10}(+1)^{15\sqrt{n}},y^{10}(+1)^{15\sqrt{n}}\right).\end{aligned}$$

Note that due to the symmetry of the *gap Hamming distance* problem, a protocol for $\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}$ implies a protocol for $\neg\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}$. Hence, if we assume by contradiction that there exists an $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for $\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}$, where $0<\epsilon<\frac{1}{4}$ and $T\cdot W=o(n)$ (which in turn implies that there exists an $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for $\neg\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}$, where $0<\epsilon<\frac{1}{4}$ and $T\cdot W=o(n)$), then by applying Lemma 5.8 we get an $\mathcal{MA}_{2\epsilon}(T,W)$ communication complexity protocol for $\mathsf{ORT}_{n,2\sqrt{n},\sqrt{n}}(x,y)$ such that $T\cdot W=o(n)$, in contradiction to Corollary 5.7. Thus we get the following corollary,

Corollary 5.9. *Let ϵ be a positive constant, such that $\epsilon<\frac{1}{4}$. For every $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for $\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}$ we have $T\cdot W=\Omega(n)$, hence $T+W=\Omega(\sqrt{n})$.*

Finally, we note that in previous work [CR11] provided a toolkit of simple reductions that can be used to generalize a lower bound on the communication complexity of *gap Hamming distance* for every reasonable parameter settings. Specifically, a lower bound for $\mathsf{GHD}_{10n+15\sqrt{n},\sqrt{n},\sqrt{n}}$ implies a lower bound for $\mathsf{GHD}=\mathsf{GHD}_{n,\sqrt{n},\sqrt{n}}$. Moreover, we note that their reduction is directly robust for $\mathcal{MA}$ communication complexity; thus we conclude,

Theorem 5.10. *Let ϵ be a positive constant, such that $\epsilon<\frac{1}{4}$. For every $\mathcal{MA}_\epsilon(T,W)$ communication complexity protocol for GHD we have $T\cdot W=\Omega(n)$, hence $T+W=\Omega(\sqrt{n})$.*

5.2 Upper bound

In their seminal paper, Aaronson and Widgerson [AW09] showed an $\mathcal{MA}$ communication complexity protocol for the disjointness problem, wherein the communication complexity is $O(\sqrt{n}\log n)$, and the size of the proof is also $O(\sqrt{n}\log n)$.

We modify their protocol in order to show an $\mathcal{MA}$ communication complexity protocol for GHD, wherein the communication complexity is $O(T\log n)$, and the size of the proof is $O(W\log n)$, for every $T\cdot W\geq n$.

Theorem 5.11. *Let $T,W\in\mathbb{N}$ such that $T\cdot W\geq n$. Then, there exists an explicit $\mathcal{MA}_{1/3}(T\log n,W\log n)$ communication complexity protocol for GHD.*

Proof. Let $T,W\in\mathbb{N}$ such that $T\cdot W\geq n$. Assume for simplicity and without loss of generality that $T\cdot W=n$ exactly. Let $a\coloneqq(a_1,\ldots,a_n)\in\{-1,1\}^n$ be the input of Alice, and $b\coloneqq(b_1,\ldots,b_n)\in\{-1,1\}^n$ be the input of Bob. Let each player define a bivariate function that represents its input; more precisely, let Alice define $f_a:[W]\times[T]\to\{-1,1\}$ by

$$f_a(x,y)=a_{(x-1)T+y},$$

and similarly, let Bob define $f_b:[W]\times[T]\to\{-1,1\}$ by

$$f_b(x,y)=b_{(x-1)T+y}.$$

Fix a prime $q \in [6n, 12n]$. Note that f_a and f_b have unique extensions $\tilde{f}_a : \mathbb{F}_q^2 \to \mathbb{F}_q$ and $\tilde{f}_b : \mathbb{F}_q^2 \to \mathbb{F}_q$ (respectively) as polynomials of degree $(W-1)$ in the first variable, and degree $(T-1)$ in the second variable. Next, define the polynomial $s : \mathbb{F}_q \to \mathbb{F}_q$ by

$$s(x) = \sum_{y \in [T]} \tilde{f}_a(x, y)\tilde{f}_b(x, y).$$

Note that the degree of s is at most $2(W-1)$. Denote the Hamming distance of a and b by $\mathsf{HD}(a, b)$. Then,

$$\mathsf{HD}(a, b) = \frac{n - \sum_{x \in [W]} s(x)}{2}. \tag{5.6}$$

Thus, it is sufficient for one of the players to know s in order to compute the Hamming distance. We define the following $\mathcal{MA}$ communication complexity protocol:

1. Merlin sends Alice a message that consists of the coefficients of a polynomial $s' : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most $2(W-1)$, for which Merlin claims that $s' = s$.
2. Bob uniformly picks $r \in \mathbb{F}_q$, and sends Alice a message that consists of r and
$$\tilde{f}_b(r, 1), \ldots, \tilde{f}_b(r, T).$$
3. Alice computes $s(r) = \sum_{y \in [T]} \tilde{f}_a(r, y)\tilde{f}_b(r, y)$ and $s'(r)$. If $s(r) = s'(r)$, Alice computes $\mathsf{HD}(a, b) = \frac{n - \sum_{x \in [W]} s'(x)}{2}$ and returns the result. Otherwise, Alice rejects the proof and returns $\perp$.

Figure 2: $\mathcal{MA}$ Communication Complexity Protocol for GHD

Note that Merlin sends the coefficients of a polynomial of degree at most $2(W-1)$ over a finite field of cardinality $O(n)$. Hence the size of the proof is $O(W \log n)$. In addition, note that the entire communication between Alice and Bob consists of sending the element r and the T evaluations of $\tilde{f}_b$ (in step 2 of the algorithm). Hence the total communication complexity is $O(T \log n)$.

If Merlin is honest, then Alice can directly compute $\mathsf{HD}(a, b)$ with probability 1, as according to (5.6) the Hamming distance of a and b can be inferred from s. Otherwise, if $s' \neq s$ then by the Schwartz-Zippel Lemma

$$\Pr[s(r) = s'(r)] \leq \frac{2(W-1)}{q} \leq \frac{1}{3},$$

(where the probability is taken over the random selection of $r \in \mathbb{F}_q$). Thus the test fails with probability at least 2/3. □

6 The AM Streaming Complexity of *Distinct Elements*

In this section we show an application of the canonical $\mathcal{AM}$ streaming algorithm for the *Distinct Elements* problem. In the regular data stream model (without any probabilistic proof system), it is well known (cf. [Mut05]) that the space complexity of the *Distinct Elements* problem is lower bounded by the size of the alphabet of the data stream (for sufficiently long data streams). In contrast, using the canonical $\mathcal{AM}$ streaming algorithm we show that by allowing $\mathcal{AM}$ proofs, we can obtain a tradeoff between the space complexity and the size of the proof.

Furthermore, we then rely on our lower bound on the $\mathcal{MA}$ communication complexity of the GHD problem, in order to show a matching lower bound on the $\widehat{\mathcal{MA}}$ streaming complexity of *Distinct Elements*.

6.1 Upper Bound

We show that for every $s, w \in \mathbb{N}$ such that $s \cdot w \geq n$ (where n is the size of the alphabet) there exists an $\mathcal{AM}$ streaming algorithm for the *Distinct Elements* problem that uses a proof of size $\tilde{O}(w)$ and a space complexity $\tilde{O}(s)$. For example, by fixing $w = n$, we have an $\mathcal{AM}$ streaming algorithm for the *Distinct Elements* problem that uses only a polylogarithmic (in the size of the alphabet and the length of the stream) number of bits of space.

Formally, we show:

Theorem 6.1. *For every $s, w \in \mathbb{N}$ such that $s \cdot w \geq n$, there exists an explicit $\mathcal{AM}_{0,1/3}\big(s \cdot \text{polylog}(m,n),\ w \cdot \text{polylog}(m,n)\big)$ streaming algorithm for the* Distinct Elements *problem, given a data stream $\sigma = (a_1, \ldots, a_m)$ with alphabet $[n]$.*

The idea behind the proof of Theorem 6.1 is simply noting that we can indicate whether an element j appears in the stream, by the disjunction of the element indicators of $j \in [n]$ in all of the positions of the stream (i.e., $\chi_1(j), \ldots, \chi_m(j)$). Then we can represent the number of distinct elements as a sum of disjunctions, and use the canonical $\mathcal{AM}$ streaming algorithm in order to solve the *Distinct Elements* problem. Formally,

Proof. Recall that the *Distinct Elements* problem is the data stream problem of computing (exactly) the following function:

$$F_0(\sigma) = \left|\{i \in [n] \ : \ \exists j \in [m] \ \ a_j = i\}\right|.$$

Observe that for every data stream we can write $F_0(\sigma)$ as

$$\sum_{j=1}^{n} \big(\chi_1(j) \vee \chi_2(j) \vee \ldots \vee \chi_m(j)\big).$$

Let $\sigma = (a_1, \ldots, a_m)$ be a data stream with alphabet $[n]$. Let $s, w \in \mathbb{N}$ such that $s \cdot w \geq n$, let $\epsilon = 0$, and let $\delta = 1/3$. By Theorem 4.1 we have an explicit $\mathcal{AM}_{\epsilon,\delta}(S, W)$-streaming algorithm for computing $F_0(\sigma)$, where $S = O\big(s \cdot \text{polylog}(m,n)\big)$ and $W = O\big(w \cdot \text{polylog}(m,n)\big)$. □

6.2 Lower bound

In the rest of this section we consider the $\widehat{\mathcal{MA}}$ model. As we mentioned in Section 3 the $\widehat{\mathcal{MA}}$ model, wherein the verifier has a free access to the proof, is stronger than the $\mathcal{MA}$ model, wherein the proof is being streamed. Hence the lower bound we prove holds for both models.

As implicitly shown in [IW03], the communication complexity problem of GHD reduces to the data stream problem of *Distinct Elements*. We note that the foregoing reduction can be adapted in order to reduce the $\mathcal{MA}$ communication complexity problem of GHD to the $\widehat{\mathcal{MA}}$ problem of approximating the number of distinct elements in a stream within a multiplicative factor of $1 \pm 1/\sqrt{n}$. Together with our lower bound on the $\mathcal{MA}$ communication complexity of GHD, this implies the following:

Theorem 6.2. *Let $\delta < \frac{1}{4}$. For every $\widehat{\mathcal{MA}}_{\frac{1}{\sqrt{n}},\delta}(S, W)$ streaming algorithm for approximating the number of distinct elements in a data stream $\sigma = (a_1, \ldots, a_m)$ (over alphabet $[n]$) we have $S \cdot W = \Omega(n)$, hence $S + W = \Omega(\sqrt{n})$.*

Proof. Let Alice hold a string $x \in \{-1, 1\}^n$ and Bob hold $y \in \{-1, 1\}^n$. Alice can convert her string $x = (x_1, \ldots, x_n)$ to a data stream over the alphabet $\Sigma = \big\{(i, b) \mid i \in [n],\ b \in \{-1, 1\}\big\}$ in the following manner:

$$\sigma_A = \big((1, x_1), (2, x_2), \ldots, (n, x_n)\big).$$

Similarly, Bob can convert $y = (y_1, \ldots, y_n)$ to

$$\sigma_B = \big((1, y_1), (2, y_2), \ldots, (n, y_n)\big).$$

Observe that all of the elements in σ_A are distinct, and that all of the elements in σ_B are also distinct. In addition, note that the only way in which an element can appear twice in the concatenation of the streams is if $x_i = y_i$ for some $i \in [n]$. In fact, if we denote the number of distinct elements in $\sigma_A \circ \sigma_B$ by d, and denote the Hamming distance of x and y by $\mathsf{HD}(x, y)$, then we have the following relation:

$$d = n + \mathsf{HD}(x, y). \tag{6.1}$$

Alice and Bob can simulate running an $\widehat{\mathcal{MA}}$ streaming algorithm on the concatenation of their inputs by using a one-way $\mathcal{MA}$ communication complexity protocol, such that the number of the bits that are being communicated during the execution of the protocol is exactly the same as the number of bits of space that are used by the simulated $\widehat{\mathcal{MA}}$ streaming algorithm. Details follow.

Say we have an $\widehat{\mathcal{MA}}_{\frac{1}{\sqrt{n}},\delta}(S, W)$ streaming algorithm $\mathcal{A}$ for approximating the number of distinct elements in $\sigma_A \circ \sigma_B$. Alice can run $\mathcal{A}$ on σ_A, using a proof w of size W. After the algorithm finished processing the last element of σ_A, Alice sends the current state of her memory (which consists of at most S bits) to Bob. Next, Bob sets his memory to the state

that Alice had sent, uses the proof w, and completes the run of $\mathcal{A}$ over σ_B. Note that the total communication during the execution of the aforementioned protocol is at most S bits, as the data stream algorithm uses at most S bits of space during its execution.

As a conclusion, if there exists such $\mathcal{A}$ then by the reduction above there exists an $\mathcal{MA}$ communication protocol that outputs a $1 \pm 1/\sqrt{n}$ multiplicative approximation of d. By (6.1) we can compute $\widetilde{\mathsf{HD}}(x, y)$, such that

$$\mathsf{HD}(x,y) - \sqrt{n} - \frac{\mathsf{HD}(x,y)}{\sqrt{n}} < \widetilde{\mathsf{HD}}(x,y) < \mathsf{HD}(x,y) + \sqrt{n} + \frac{\mathsf{HD}(x,y)}{\sqrt{n}},$$

or

$$\mathsf{HD}(x,y) - 2\sqrt{n} < \widetilde{\mathsf{HD}}(x,y) < \mathsf{HD}(x,y) + 2\sqrt{n}.$$

Thus we can solve $\mathsf{GHD}_{n,2\sqrt{n},2\sqrt{n}}$ while communicating at most $O(S)$ bits and using a proof of at most $O(W)$ bits. Hence, using the toolkit of reductions provided in [CR11] (see Section 5.1) we can solve GHD, while communicating at most $O(S)$ bits and using a proof of at most $O(W)$ bits. Thus, by Theorem 5.10 we have $S \cdot W = \Omega(n)$, hence $S + W = \Omega(\sqrt{n})$. □

Note that in particular, Theorem 6.2 implies a lower bound (with the same parameters) on the $\widehat{\mathcal{MA}}$ streaming complexity of computing the *exact* number of distinct elements in a stream.

Last, we also note that by a straightforward adaptation of the reduction from the communication complexity problem of GHD to the data stream problem of *Empirical Entropy* (see [CCM07]), our $\mathcal{MA}$ lower bound on GHD also implies an $\widehat{\mathcal{MA}}$ lower bound on the *Empirical Entropy* problem.

References

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 20–29, New York, NY, USA, 1996. ACM.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1:2:1–2:54, February 2009.

[BC09] Joshua Brody and Amit Chakrabarti. A multi-round communication lower bound for gap hamming and some consequences. In *IEEE Conference on Computational Complexity*, pages 358–368, 2009.

[BHR+07] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. On synopses for distinct-value estimation under multiset operations. In *SIGMOD Conference*, pages 199–210, 2007.

[BYJK+02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM*, pages 1–10, 2002.

[CBM06] Amit Chakrabarti, Khanh Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. In *In Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS*. Springer, 2006.

[CCM07] Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. A near-optimal algorithm for computing the entropy of a stream. In *In ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335, 2007.

[CCM09] Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. Annotations in data streams. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 222–234, Berlin, Heidelberg, 2009. Springer-Verlag.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In *CRYPTO*, pages 151–168, 2011.

[CMT10] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *CoRR*, abs/1004.2899, 2010.

[CMT11] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. *CoRR*, May 2011.

[CR11] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 51–60, New York, NY, USA, 2011. ACM.

[FM83] Philippe Flajolet and G. Nigel Martin. Probabilistic counting. In *FOCS*, pages 76–82, 1983.

[GKG05] Sumit Ganguly, Iit Kanpur, and Minos Garofalakis. Join-distinct aggregate estimation over update streams. In *In Proc. ACM PODS*, 2005.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[IW03] Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *FOCS*, pages 283–, 2003.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52, 2010.

[Mut05] S. Muthukrishnan. *Data streams: algorithms and applications*. Now Publishers, 2005.

[Raz87] A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. Notes of the Academy of Science of the USSR: 41(4) : 333-338, 1987.

[RS04] Ran Raz and Amir Shpilka. On the power of quantum proofs. *Computational Complexity, Annual IEEE Conference on*, 0:260–274, 2004.

[She11] Alexander A. Sherstov. The communication complexity of gap hamming distance. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:63, 2011.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 77–82, New York, NY, USA, 1987. ACM.

[Vid11] Thomas Vidick. A concentration inequality for the overlap of a vector on a large set, with application to the communication complexity of the gap-hamming-distance problem. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:51, 2011.

[Yao83] Andrew C. Yao. Lower bounds by probabilistic arguments. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, pages 420–428, Washington, DC, USA, 1983. IEEE Computer Society.